IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 19, 2021, accepted February 27, 2021, date of publication March 2, 2021, date of current version March 11, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3063669*

# TWINBOT: Autonomous Underwater Cooperative Transportation

**ROGER PI[1], PATRYK CIEŚLAK[1], PERE RIDAO[1], (Member, IEEE), AND PEDRO J. SANZ[2], (Senior Member, IEEE)**

[1]Computer Vision and Robotics Research Institute (VICOROB), Universitat de Girona, 17003 Girona, Spain
[2]Interactive and Robotic Systems Lab (IRS-Lab), Jaume I University, 12071 Castellón de la Plana, Spain

Corresponding author: Roger Pi (roger.pi@udg.edu)

This work was supported in part by the TWINBOT ”TWIN ROBOTS FOR COOPERATIVE UNDERWATER INTERVENTION MISSIONS” Project funded by the Spanish Ministry of economy, industry, and competitiveness, under Grant DPI2017-86372-C3, in part by the Valencian Government (CIRTESU Project) under Grant IDIFEDER/2018/013, and in part by the Secretaria d'Universitats i Recerca del Departament d'Economia i Coneixement de la Generalitat de Catalunya under Grant 2019FI_B_00812.

**ABSTRACT** Underwater Inspection, Maintenance, and Repair operations are nowadays performed using Remotely Operated Vehicles (ROV) deployed from dynamic-positioning vessels, having high daily operational costs. During the last twenty years, the research community has been making an effort to design new Intervention Autonomous Underwater Vehicles (I-AUV), which could, in the near future, replace the ROVs, significantly decreasing these costs. Until now, the experimental work using I-AUVs has been limited to a few single-vehicle interventions, including object search and recovery, valve turning, and hot stab operations. More complex scenarios usually require the cooperation of multiple agents, i.e., the transportation of large and heavy objects. Moreover, using small, autonomous vehicles requires consideration of their limited load capacity and limited manipulation force/torque capabilities. Following the idea of multi-agent systems, in this paper we propose a possible solution: using a group of cooperating I-AUVs, thus sharing the load and optimizing the stress exerted on the manipulators. Specifically, we tackle the problem of transporting a long pipe. The presented ideas are based on a decentralized Task-Priority kinematic control algorithm adapted for the highly limited communication bandwidth available underwater. The aforementioned pipe is transported following a sequence of poses. A path-following algorithm computes the desired velocities for the robots' end-effectors, and the on-board controllers ensure tracking of these setpoints, taking into account the geometry of the pipe and the vehicles' limitations. The utilized algorithms and their practical implementation are discussed in detail and validated through extensive simulations and experimental trials performed in a test tank using two 8 DOF I-AUVs.

**INDEX TERMS** Autonomous underwater intervention, cooperative robots, cooperative manipulation, task priority control.

## I. INTRODUCTION

Inspection, Maintenance, and Repair(IMR) operations at sea remain extremely costly and time-consuming, requiring the use of heavy-weight Remotely Operated Vehicles (ROV) supported by large Dynamic Positioning (DP) vessels and complex Tether Management Systems (TMS). In the last three decades, research in autonomous underwater robots and robotic intervention has been slowly gaining speed, aiming to tackle some of the IMR tasks that could in future be performed by Intervention Autonomous Underwater Vehicles (I-AUV). The research started with the pioneering works of OTTER [1], ODIN [2], UNION [3] and AMADEUS [4], which contributed in developing core technologies. The first field demonstrations of actual autonomous IMR operation arrived with the ALIVE project [5], where an I-AUV docked with a subsea control panel using hydraulic grasps and performed fixed-base valve turning. Similar experiments were reproduced later in the TRITON [6] project, using a significantly lighter I-AUV. Valve turning using a floating I-AUV was demonstrated during the PANDORA European project, first using learning-by-demonstration techniques [7] and later using Task Priority (TP) [8]. Valve turning in the presence of obstacles was tackled in [9] using the TP framework

The associate editor coordinating the review of this manuscript and approving it for publication was Cheng Chin.

and assuming a prior knowledge of the obstacles' positions. In [10], a method is presented employing laser scanning to build an occupancy grid used for motion planning. That paper reports experimental results obtained on autonomous manipulation in the presence of *a priori* unknown obstacles. Another typical IMR task already demonstrated is pipe inspection [11]. In this case, compliant control methods were used to ensure robust contact between the inspection tool and the inspected surface.

Object search and recovery has been another area of I-AUV research. The first object recovery from a floating vehicle was achieved in the SAUVIM [12] project. The I-AUV autonomously located and hooked an *a priori* known object, while hovering to recover it from the seabed. Similar results were recently reported in the MARIS project [13], where an I-AUV was able to detect and grasp a small pipe from the bottom of a water tank, using a specially designed 3-finger hand and a single 7 DOF robotic arm. A multipurpose object search and recovery strategy was first proposed in a Spanish project RAUVI [14], extended later in the European project TRIDENT [13], [15]. This strategy was organized in two steps: 1) First, the I-AUV performed an optical survey of the area of interest, building a photo-mosaic, and 2) The user selected a target object in the photo-mosaic and the robot was sent to recover it autonomously.

The use of multiple robots is a natural step forward in the research on autonomous underwater manipulation. Small robots do not require DP vessels and can be deployed from virtually any ship, significantly reducing the operational costs. Nevertheless, such small vehicles have limited capabilities in terms of their load capacity and the magnitude of the forces/torques that they can exert. A multi-robot system can share the load and optimize the manipulator's stress, allowing for manipulation of long, bulky, and/or heavy objects. Additional benefits of a multi-robot system are its increased robustness and coverage. In the presence of group redundancy, it may be possible to complete a task even if one of the robots fails, especially if there is no distinctive leader and the group is homogeneous [16]. On the other hand, better efficiency and shorter mission time can be achieved in search and recovery scenarios. The obvious drawback of a multi-robot system is the need to develop and implement more complex control algorithms, able to deal with the coordination of robots, taking into account the available communication links.

## A. RELATED WORK

A recent survey of multi-robot manipulation, focused on cooperative transportation using ground robots, is presented in [16]. In the case of underwater robots, we are naturally interested in the concepts developed for grasping-based transportation. Very few works exist in this field. After completing the TRIDENT project in 2013, our consortium developed the idea of evolving its concept into a complete cooperative system involving two cooperative I-AUVs for load transportation and object assembly: the "Cooperative Robots for Autonomous Underwater Intervention Missions" (CRAUNIM) concept. Although this proposal did not mature at the EU level, it was later developed in the MARIS and TWINBOT national projects. MARIS was an Italian project [17] which experimentally demonstrated single-vehicle floating-base object recovery, as well as the kinematic simulation of a cooperative load transportation task. On the other hand, TWINBOT is a Spanish project devoted to the experimental demonstration of cooperative transportation. Simulated results of underwater cooperative object transportation have been reported in [18] and [19]. In the first case, the MARIS control strategy was based on the TP kinematic control algorithm, running independently on two I-AUVs, cooperating to transport a rigid pipe. First, each robot computes its optimal End-Effector (EE) velocity. Later, a consensus is reached, i.e., an average pipe velocity is computed, for which the robots recalculate their configuration space velocities. The second investigation treats the robots as particles immersed in a potential field generated by the goal (attracting field) and the obstacles (repelling field). Only simulation results about formation control, including I-AUV-object and I-AUV-object-environment interactions, are reported. To the authors' knowledge, there is only one previous study reporting experimental results on cooperative transportation underwater, which was presented in [20]. There, a decentralized impedance control method is proposed, where each robot is equipped with a wrist-mounted force-torque sensor. The control strategy is based on the leader-follower concept, where the leader knows the environment and commands the motion of the transported object. The followers estimate the object's motion using the force and torque readings. All robots implement an impedance control strategy based on their dynamical models.

## B. CONTRIBUTION

This paper presents simulation and experimental results of the TWINBOT project, demonstrating the complete cooperative transportation sequence of pick, transport, and placement of a bulky object. The main contributions of this paper are:

1) The proposal of a simple decentralized TP kinematic control architecture, using a master-slave organization, suitable for the limited available communication bandwidth.
2) The proposal of a distributed velocity normalization method to accomplish the velocity limits among all the I-AUVs, which are required to ensure the direction of the desired object's velocity.
3) The use of a high accuracy, drift-less, vision-based navigation, using an *a priori* known optical map.
4) The use of a visual servoing method to achieve the accuracy required for grasping.
5) Experimental validation in a water tank environment.

The choice of the TP kinematic control, for solving the cooperative transportation problem, connects our work with the MARIS project, where a similar approach was used.

Therefore, it is important to understand the differences between the algorithm used in MARIS and the one proposed in this work.

In MARIS, the transportation phase requires three steps: 1) Independent optimization of full TP hierarchies for each of the robots separately, with safety tasks at the top, based on the same desired EE velocity, 2) Averaging of velocities achievable by the systems, 3) Second independent optimization of TP hierarchies with the EE velocity tracking tasks as the highest priority. As the authors underline, this strategy leads to the best effort solution, because in step 2 of the algorithm a simple average is used, which does not guarantee that the resulting desired EE velocity can be achieved in step 3. The authors provide a discussion on how a more advanced fusion policy may solve the problem, but they leave the actual derivation of such a policy for future work.

In the present work, the transportation phase requires only two steps: 1) Independent optimization of the full TP hierarchies for each of the robots separately, with safety tasks at the top, based on the EE velocities required to move a transported pipe to a specific position with a specific orientation, 2) Normalization of the resulting velocities which ensures that each of the robots can achieve EE velocity in all directions (linear and angular). While the first step is equivalent in both algorithms, our velocity normalization (step 2) allows solving the problem with one TP computation only, ensuring that the desired EE velocities are possible to achieve while satisfying all the safety tasks.

## II. UVMS KINEMATICS

In this section, we present the kinematics of an Underwater Vehicle Manipulator System (UVMS). First, we define its generalized coordinates and then develop the position and velocity kinematics.

### A. GENERALIZED COORDINATES

The generalised coordinates of an I-AUV system can be defined as follows:

$$\mathbf{q} = [\boldsymbol{\eta}\ \boldsymbol{q}]^T \tag{1}$$

where $\boldsymbol{\eta}$ is the generalized coordinates vector of the AUV (pose vector) and $\boldsymbol{q}$ is the generalized coordinate vector of the arm (configuration vector):

$$\boldsymbol{\eta} = [\boldsymbol{\eta}_1^T\ \boldsymbol{\eta}_2^T]^T = [x\ y\ z\ \phi\ \theta\ \psi]^T \tag{2}$$

$$\boldsymbol{q} = [q_1 \dots q_n]^T. \tag{3}$$

### B. KINEMATICS OF POSITION

The position and attitude of the AUV *B–frame* with respect to the North East Down (NED) *N–frame* can be represented using the pose vector $\boldsymbol{\eta}$ or the related homogeneous matrix:

$$\begin{aligned} {}^N\boldsymbol{K}_B(\boldsymbol{\eta}) &= \begin{bmatrix} {}^N\boldsymbol{R}_B(\boldsymbol{\eta}_2) & \boldsymbol{\eta}_1 \\ 0_{1\times 3} & 1 \end{bmatrix} \\ {}^N\boldsymbol{R}_B(\boldsymbol{\eta}_2) &= \boldsymbol{R}_{z,\psi}\boldsymbol{R}_{y,\theta}\boldsymbol{R}_{x,\phi}. \end{aligned} \tag{4}$$

The position and attitude of the arm, with respect to the 0*–frame* located at its base, can be represented with the pose vector:

$${}^0\boldsymbol{\eta}_n = [{}^0\boldsymbol{\eta}_{0n_1}{}^T\ {}^0\boldsymbol{\eta}_{0n_2}{}^T]^T = [x_{0n}\ y_{0n}\ z_{0n}\ \phi_{0n}\ \theta_{0n}\ \psi_{0n}]^T \tag{5}$$

which can be computed from the arm configuration vector ($\boldsymbol{q}$) computing the forward kinematics of the arm [21]:

$${}^0\boldsymbol{A}_n(\boldsymbol{q}) = \prod_{i=1}^{n} {}^{i-1}\boldsymbol{A}_i(\boldsymbol{q}_i) = \begin{bmatrix} {}^0\boldsymbol{R}_n({}^0\boldsymbol{\eta}_{0n_2}) & {}^0\boldsymbol{\eta}_{0n_1} \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{6}$$

where ${}^{i-1}\boldsymbol{A}_i(\boldsymbol{q}_i)$ are the link to link transformation matrices depending on the *DH* parameters. The Euler angles of the arm ($\boldsymbol{\eta}_{0n_2} = [\phi_{0n}\ \theta_{0n}\ \psi_{0n}]^T$) can be obtained from:

$${}^0\boldsymbol{R}_n({}^0\boldsymbol{\eta}_{0n_2}) = \boldsymbol{R}_{z,\psi_{0n}}\boldsymbol{R}_{y,\theta_{0n}}\boldsymbol{R}_{x,\phi_{0n}} \tag{7}$$

solving for $\phi_{0n}$, $\theta_{0n}$ and $\psi_{0n}$.

Therefore, given the UVMS generalized coordinates $\mathbf{q}$, its end-effector pose:

$$\boldsymbol{\eta}_{ee} = [\boldsymbol{\eta}_{ee_1}\ \boldsymbol{\eta}_{ee_2}]^T = [x_{ee}\ y_{ee}\ z_{ee}\ \phi_{ee}\ \theta_{ee}\ \psi_{ee}]^T \tag{8}$$

defined with respect to *N–frame*, depends on the AUV pose $\boldsymbol{\eta}$ and the end-effector pose ${}^0\boldsymbol{\eta}_{0n}$ (Fig. 1). The UVMS pose, can also be expressed as a homogeneous matrix:

$$\begin{aligned} {}^N\boldsymbol{K}_n(\mathbf{q}) &= {}^N\boldsymbol{K}_B(\boldsymbol{\eta}) \cdot {}^B\boldsymbol{H}_0 \cdot {}^0\boldsymbol{A}_n(\boldsymbol{q}) \\ &= \begin{bmatrix} {}^N\boldsymbol{R}_n(\boldsymbol{\eta}_{ee_2}) & \boldsymbol{\eta}_{ee_1} \\ 0_{1\times 3} & 1 \end{bmatrix}. \end{aligned} \tag{9}$$

where ${}^B\boldsymbol{H}_0$ is the transformation matrix from the *B–frame* to the 0*–frame*.

### C. KINEMATICS OF VELOCITY

The UVMS Jacobian [22] relates the quasi-velocities $\boldsymbol{\zeta} = [\boldsymbol{\nu}^T\ \dot{\boldsymbol{q}}^T]^T$ to the end-effector rate of change $\dot{\boldsymbol{\eta}}_{ee}$:

$$\dot{\boldsymbol{\eta}}_{ee} = \boldsymbol{J}(\mathbf{q})\boldsymbol{\zeta} \tag{10}$$

being $\boldsymbol{\nu} = [\boldsymbol{\nu}_1{}^T\ \boldsymbol{\nu}_2{}^T]^T = [u\ v\ w\ p\ q\ r]^T$ the AUV linear and angular velocity vector. Given the AUV Jacobian:

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{J}_\nu(\boldsymbol{\eta})\boldsymbol{\nu} \\ \boldsymbol{J}_\nu(\boldsymbol{\eta}) &= \begin{bmatrix} {}^N\boldsymbol{R}_B(\boldsymbol{\eta}_2) & 0_{3\times 3} \\ 0_{3\times 3} & \boldsymbol{J}_{\nu_2}(\boldsymbol{\eta}_2) \end{bmatrix}, \end{aligned} \tag{11}$$

where $\boldsymbol{J}_{\nu_2}(\boldsymbol{\eta}_2)$ is the matrix mapping the angular velocity into the Euler angle rates, and given the arm geometric Jacobian:

$$\begin{aligned} \begin{bmatrix} {}^0\dot{\boldsymbol{\eta}}_{0n_1} \\ {}^0\boldsymbol{\nu}_{0n_2} \end{bmatrix} &= \boldsymbol{J}_m(\boldsymbol{q})\dot{\boldsymbol{q}} \\ \boldsymbol{J}_m(\boldsymbol{q}) &= \begin{bmatrix} \boldsymbol{J}_{m,p}(\boldsymbol{q}) \\ \boldsymbol{J}_{m,o}(\boldsymbol{q}) \end{bmatrix} \\ \boldsymbol{J}_{m,p}(\boldsymbol{q}) &= \frac{\partial \boldsymbol{\eta}_{0n_1}}{\partial \boldsymbol{q}} \\ \boldsymbol{J}_{m_o}(\boldsymbol{q}) &= \frac{\partial \boldsymbol{\nu}_{0n_2}}{\partial \boldsymbol{q}}, \end{aligned} \tag{12}$$

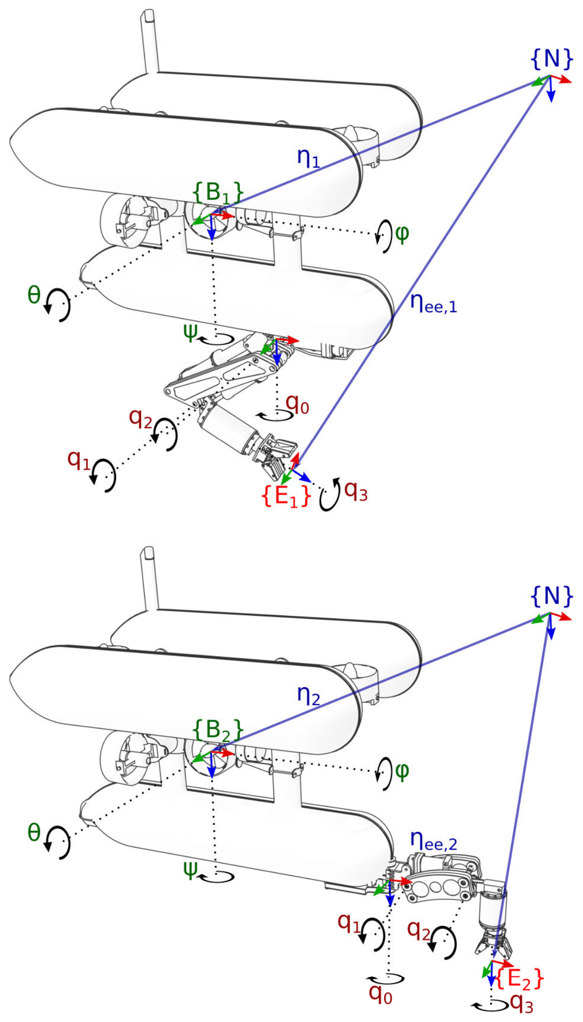


**FIGURE 1.** Kinematic structure of the UVMS team.

the UVMS geometric Jacobian $\boldsymbol{J}$ (10) can be computed as follows:

$$\begin{bmatrix}\dot{\boldsymbol{\eta}}_{\boldsymbol{ee}_1}\\ \boldsymbol{\nu}_{\boldsymbol{ee}_2}\end{bmatrix}=\begin{bmatrix}\boldsymbol{J_p}(\mathbf{q})\\ \boldsymbol{J_o}(\mathbf{q})\end{bmatrix}\begin{bmatrix}\boldsymbol{\nu}\\ \dot{\boldsymbol{q}}\end{bmatrix}=\boldsymbol{J}(\mathbf{q})\boldsymbol{\zeta} \tag{13}$$

where

$$\begin{aligned}\boldsymbol{J_p}(\mathbf{q})&=\left[{}^N\boldsymbol{R_B}\ -\boldsymbol{H}{}^N\boldsymbol{R_B}\ \boldsymbol{J_{m,p}(q)}\right]\\ \boldsymbol{H}&=[{}^N\boldsymbol{R_B}{}^B\boldsymbol{r_{B0}}]_\times+[{}^N\boldsymbol{R_0}{}^0\boldsymbol{\eta_{0,ee}}]_\times\end{aligned} \tag{14}$$

and

$$\boldsymbol{J_o}(\mathbf{q})=\left[0_{3\times3}\ {}^N\boldsymbol{R_B}\ \boldsymbol{J_{m,o}(q)}\right], \tag{15}$$

being $[\boldsymbol{a}]_\times$ the skew symmetric matrix so that $\boldsymbol{c}=\boldsymbol{a}\times\boldsymbol{b}=[\boldsymbol{a}]_\times\boldsymbol{b}$.

## III. TASK-PRIORITY CONTROL

Tasks are designed to achieve goals, such as reaching an EE pose or avoiding joint limits. Exploiting the system redundancy, several tasks may run concurrently satisfying their goals simultaneously, e.g., achieving an EE pose while remaining within the joint limits. The priorities are used to define a strict hierarchy, stating, in case of conflicts among tasks' goals, which ones must be respected and which ones can be sacrificed. Two categories of tasks can be identified: 1) Equality tasks, whose goal is to drive the task variable to a desired value (e.g., EE pose task) and 2) Set tasks (also called inequality tasks) [23], [24], whose goal is to keep the task variable within a specified range (e.g., manipulator joint limits task).

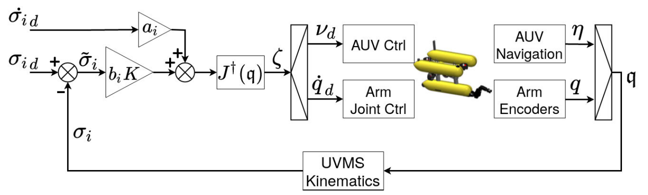


**FIGURE 2.** Kinematic control for a general task $\dot{\sigma}_i$ according to (20).

### A. TASK DEFINITION

A task is a m-dimensional functional constraint defined over the generalised coordinates $\mathbf{q}=[\boldsymbol{\eta}^T\ \boldsymbol{q}^T]^T=[x\ y\ z\ \phi\ \theta\ \psi\ q_1\ \ldots\ q_n]^T$:

$$\boldsymbol{\sigma_i}\equiv\boldsymbol{\sigma_i}(\mathbf{q})\in\mathbb{R}^{\boldsymbol{m_i}}, \tag{16}$$

whose time derivative ($\dot{\boldsymbol{\sigma}}_{\boldsymbol{i}}$) is related to the system quasi-velocities ($\boldsymbol{\zeta}=[\boldsymbol{\nu}^T\ \dot{\boldsymbol{q}}^T]^T=[u\ v\ w\ p\ q\ r\ q_1\ \ldots\ q_n]^T$) through the corresponding Jacobian $\boldsymbol{J_i}(\mathbf{q})$ :

$$\dot{\boldsymbol{\sigma}}_{\boldsymbol{i}}(\mathbf{q})=\boldsymbol{J_i}(\mathbf{q})\boldsymbol{\zeta}, \tag{17}$$

where

$$\boldsymbol{J_i}(\mathbf{q}))=\frac{\partial\boldsymbol{\sigma_i}(\mathbf{q})}{\partial\mathbf{q}}\frac{\partial\mathbf{q}}{\partial t}=\frac{\partial\boldsymbol{\sigma_i}(\mathbf{q})}{\partial\mathbf{q}}\begin{bmatrix}\boldsymbol{J_v}(\boldsymbol{\eta})&\boldsymbol{0_{6\times n}}\\ \boldsymbol{0_{n\times6}}&\boldsymbol{I_{n\times n}}\end{bmatrix}\boldsymbol{\zeta} \tag{18}$$

and for which a corresponding error variable is defined:

$$\tilde{\boldsymbol{\sigma}}_{\boldsymbol{i}}=\tilde{\boldsymbol{\sigma}}_{\boldsymbol{i}}(\mathbf{q})\in\mathbb{R}^{\boldsymbol{m}},\ \tilde{\boldsymbol{\sigma}}_{\boldsymbol{i}}=\boldsymbol{\sigma_{i_d}}-\boldsymbol{\sigma_i}. \tag{19}$$

The desired task velocity vector $\dot{\boldsymbol{\sigma}}_{\boldsymbol{d\,i}}$ is then defined as:

$$\dot{\boldsymbol{\sigma}}_{\boldsymbol{i}}=a_i\dot{\boldsymbol{\sigma}}_{\boldsymbol{id}}+b_i\boldsymbol{K_i}\tilde{\boldsymbol{\sigma}}_{\boldsymbol{i}}, \tag{20}$$

where $a_i\equiv a_i\in\{0,1\}$ and $b_i\equiv b_i(\mathrm{q})\in\{0,1\}$ are binary activation functions and $\boldsymbol{K_i}$ is a positive diagonal gain matrix. Equation (20) has two components: a feedback part ($\boldsymbol{K_i}\tilde{\boldsymbol{\sigma}}_{\boldsymbol{i}}$) to achieve zero regulation error and a feed-forward part ($\dot{\boldsymbol{\sigma}}_{\boldsymbol{i}}$) providing zero tracking error. Depending on the activation functions, it is possible to carry out regulation ($a_i=0, b_i=1$), tracking ($a_i=b_i=1$) or optimization ($a_i=1,\ b_i=0$).

Generic tasks can be characterized as follows:

$$Task_i(\boldsymbol{\sigma_{i_d}},\dot{\boldsymbol{\sigma}}_{\boldsymbol{i_d}},a,b)=\left\{\boldsymbol{\sigma_i}(\mathbf{q}),\tilde{\boldsymbol{\sigma}}_{\boldsymbol{i}}(\mathbf{q}),\boldsymbol{J_i}(\mathbf{q}),a_i,b_i\right\}, \tag{21}$$

where the arguments within parentheses represent time-variable parameters, whereas those within curly brackets represent arguments set up during the task instance.

### B. EQUALITY TASKS

Equality tasks include two types of tasks: Regulation tasks and Tracking tasks. The first are used to drive the task variable to the desired value and are defined by five-element tuples:

$$\begin{aligned}&Regulation\_Task_i(\boldsymbol{\sigma_{i_d}})\\&\quad = Task_i\left\{\boldsymbol{\sigma_i}(\mathbf{q}), \tilde{\boldsymbol{\sigma}}_i(\mathbf{q}), \boldsymbol{J_i}(\mathbf{q}), a_i = 0, b_i = 1\right\}.\end{aligned} \tag{22}$$

Tracking tasks are used to drive the task variable to the desired value at a certain time and are defined by seven-element tuples:

$$\begin{aligned}&Tracking\_Task_i(\boldsymbol{\sigma_{i_d}}, \dot{\boldsymbol{\sigma}}_{\boldsymbol{i_d}})\\&\quad = Task_i\left\{\boldsymbol{\sigma_i}(\mathbf{q}), \tilde{\boldsymbol{\sigma}}_i(\mathbf{q}), \boldsymbol{J_i}(\mathbf{q}), a_i = a, b_i = b\right\}.\end{aligned} \tag{23}$$

Several equality tasks have been reported in the literature to control the following: EE position, EE orientation, EE configuration (position and orientation), EE field of View, Robot Nominal Configuration, Vehicle orientation, Vehicle Yaw, and many others (see [22] for details). Hereafter, two examples are provided, which are used later on for the cooperative object transportation described in section IV.

#### 1) ROBOT NOMINAL CONFIGURATION

Regulating the configuration space variables is used for multiple tasks, including keeping a specific vehicle yaw attitude with respect to a manipulated object, optimizing the shape of the manipulator, or locking the selected manipulator joints. This task can be used, for instance, to regulate the robot heading:

$$\begin{aligned}&T_{NC\psi}(\psi_d)\\&\quad = Regulation\_Task\left\{\begin{matrix}\sigma_i(\mathbf{q}) = \psi\\ \tilde{\sigma}_i(\mathbf{q}) = \psi_d - \psi\\ \boldsymbol{J_i}(\mathbf{q}) = \begin{bmatrix}\mathbf{0}_{1\times 5} & 1 & \mathbf{0}_{\mathbf{1}\boldsymbol{xn}}\end{bmatrix}\end{matrix}\right\}\end{aligned} \tag{24}$$

or to move the arm to a desired position:

$$\begin{aligned}&T_{NC_q}(\boldsymbol{q_d})\\&\quad = Regulation\_Task\left\{\begin{matrix}\boldsymbol{\sigma_i}(\mathbf{q}) = \boldsymbol{q}\\ \tilde{\boldsymbol{\sigma}}_i(\mathbf{q}) = \boldsymbol{q_d} - \boldsymbol{q}\\ \boldsymbol{J_i}(\mathbf{q}) = \begin{bmatrix}\mathbf{0}_{1\times 6} & 1 & \dots & 1\end{bmatrix}\end{matrix}\right\}.\end{aligned} \tag{25}$$

#### 2) END-EFFECTOR CONFIGURATION

This task generates system velocities that result in following setpoints in EE velocity $\boldsymbol{\nu_{ee_d}}$, EE pose $\boldsymbol{\eta_{ee_d}}$, or EE trajectory $\boldsymbol{\eta_{ee_d}}$. The task variables are defined as follows:

$$\begin{aligned}T_{EEC}(\boldsymbol{\eta_{ee_d}}, \boldsymbol{\nu_{ee_d}}) &= Tracking\_Task\left\{\begin{matrix}\boldsymbol{\sigma_i}(\mathbf{q})\\ \tilde{\boldsymbol{\sigma}}_i(\mathbf{q})\\ \boldsymbol{J_i}(\mathbf{q})\end{matrix}\right\}\\ \boldsymbol{\sigma_i}(\mathbf{q}) &= \boldsymbol{\eta_{ee}}\\ \tilde{\boldsymbol{\sigma}}_i(\mathbf{q}) &= \begin{bmatrix}\boldsymbol{\eta_{ee_{1_d}}} - \boldsymbol{\eta_{ee_1}}\\ \lambda_{ee}\boldsymbol{\varepsilon_d} - \lambda_d\boldsymbol{\varepsilon_{ee}} + [\boldsymbol{\varepsilon_d}]_\times\boldsymbol{\varepsilon_{ee}}\end{bmatrix}\\ \boldsymbol{J_i}(\mathbf{q}) &= \boldsymbol{J}(\mathbf{q})\end{aligned} \tag{26}$$

where $\boldsymbol{J}(\mathbf{q})$ is the UVMS Jacobian (13) and $\tilde{\boldsymbol{Q}} = \{\tilde{\varepsilon}, \tilde{\lambda}\}$ is the error quaternion:

$$\begin{aligned}\tilde{\boldsymbol{Q}} &= \boldsymbol{Q_d} * \boldsymbol{Q}^{-1}\\ &= \{\lambda_{ee}\boldsymbol{\varepsilon_d} - \lambda_d\boldsymbol{\varepsilon_{ee}} + [\boldsymbol{\varepsilon_d}]_\times\boldsymbol{\varepsilon_{ee}}\ ,\ \lambda_d\lambda + \boldsymbol{\varepsilon_D^T\varepsilon}\}\end{aligned} \tag{27}$$

being $\boldsymbol{Q_d} = \{\boldsymbol{\varepsilon_d}, \lambda_d\}$ the quaternion of the desired attitude and $\boldsymbol{Q} = \{\boldsymbol{\varepsilon_{ee}}, \lambda_{ee}\}$ the one corresponding to the current attitude. It is worth noting that the direction of $\tilde{\boldsymbol{\varepsilon}}$ defines the axis of rotation between the desired and current frames. Moreover the module of the vector vanishes when aligned ($\tilde{\boldsymbol{Q}} = \{\mathbf{0}, 1\}$). Therefore, it is sufficient to use $\tilde{\boldsymbol{\varepsilon}}$ to define the attitude error. The $T_{EEC}$ task as defined above is used for EE trajectory tracking but can also be instantiated for EE configuration control alone:

$$T_{EEC_\eta}(\boldsymbol{\eta_{ee_d}}) = T_{EEC}(\boldsymbol{\eta_{ee_d}}, \mathbf{0}) \tag{28}$$

or EE velocity control alone:

$$T_{EEC_\nu}(\boldsymbol{\nu_{ee_d}}) = T_{EEC}(\mathbf{0}, \boldsymbol{\nu_{ee_d}}). \tag{29}$$

### C. SET TASKS

In this paper, we follow the implementation of the set tasks proposed in [24] where a combination of several high and low priority set tasks can be used. Set tasks are scalar regulation only tasks defined as:

$$\begin{aligned}&Set\_Task_i(\sigma_{Ui}, \sigma_{L_i}, \alpha, \delta)\\ &\quad = Task_i\left\{\begin{matrix}\sigma_i(\mathbf{q})\\ \tilde{\sigma}_i(\mathbf{q})\\ J_i(\mathbf{q})\\ \sigma_{i_d}\\ a_i\\ b_k\end{matrix}\right\}\\ a_i &= 0\\ \sigma_{i_d} &= \left\{\begin{matrix}\sigma_{L_i} + \delta, & (\sigma \leq \sigma_{L_i} + \alpha))\\ \sigma_{U_i} - \delta, & (\sigma \geq \sigma_{U_i} - \alpha)\\ 0, & otherwise \text{ (not applicable)}\end{matrix}\right\}\\ b_k &= \left\{\begin{matrix}0, & b_{k-1} \wedge ((\dot{\sigma} > 0) \wedge (\sigma \geq \sigma_{U_i} + \delta)\\ & \vee(\dot{\sigma} < 0) \wedge (\sigma \leq \sigma_{U_i} - \delta))\\ 1, & \neg b_{k-1} \wedge ((\dot{\sigma} < 0) \wedge (\sigma \leq \sigma_{L_i} + \alpha)\\ & \vee(\dot{\sigma} > 0) \wedge (\sigma \geq \sigma_{U_i} - \alpha))\\ & b_{k-1}, \quad otherwise\end{matrix}\right\}\end{aligned} \tag{30}$$

where $\sigma_{L_i}$, $\sigma_{U_i}$ can be defined as $-\infty$ or $+\infty$ respectively to define inequalities. Fig. 3 visually explains the Set Task concept. We are interested in keeping the task variable $\sigma$ within a specified range $(\sigma_L, \sigma_U)$. When $\sigma$ approaches one of the limits, at a distance $\alpha$ from the respective limit, the task is activated and starts working to push the $\sigma$ back to the desired set (top part of the figure). To avoid the chattering effect, the activated task is pushing the $\sigma$ value to reach a distance of $\delta$ from the respective limit, with $\delta > \alpha$ (bottom part of the figure). When the $\sigma$ reaches the desired set, depicted in green, the task is deactivated.

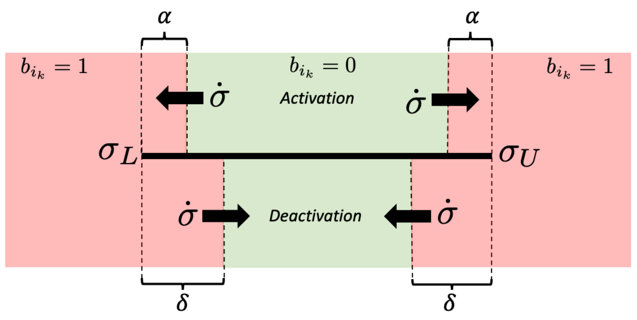


**FIGURE 3.** **Set Task concept. The task is activated when $\sigma$ approaches one of the limits, depicted in red (top), and deactivated when reaches the desired region, depicted in green (bottom).**

Typical set tasks reported in the literature include Joint Limits, EE obstacle avoidance, Minimum Altitude, and Minimum Depth, to name but a few. The most representative task of this type is probably the one devoted to ensuring that the joint variables remain within their boundaries, which is described in the next subsection.

#### 1) JOINT LIMITS

Ensuring that the control system generates position and velocity setpoints compliant with the manipulator's joint limits is a typical safety task put at the highest priority of the task hierarchy. The task is defined for each joint $j$ as follows (written for upper limit $q_{U_j}$ and lower limit $q_{L_j}$):

$$T_{JL_j}(q_{L_j}, q_{U_j}, \alpha, \delta) = Set\_Task \begin{Bmatrix} \sigma_i(\mathbf{q}) & = q_j \\ \tilde{\sigma}_i(\mathbf{q}) & = \sigma_{i_d} - \sigma_i(\mathbf{q}) \\ J_i(\mathbf{q}) = [0_0 \; \dots \; 1_j \; \dots \; 0_{6+n}] \end{Bmatrix}. \quad (31)$$

where $J_{q_j}$ is a single-entry row matrix, equal to 1 for the entry corresponding to the joint $j$. As an inequality task, its activation happens when the task variable $q_j$ is outside its boundary, where $\alpha$ and $\delta$ are the activation and deactivation margins, respectively. It is required that $\delta > \alpha$ to avoid chatter.

### D. PRIORITY-BASED EXECUTION OF MULTIPLE TASKS

Let us consider that we have a certain number of tasks: $\boldsymbol{\sigma_1}, \dots, \boldsymbol{\sigma_n}$ where, given two arbitrary tasks for which $\boldsymbol{\sigma_i}$ and $\boldsymbol{\sigma_j}$, $i < j \Rightarrow priority(\boldsymbol{\sigma_i}) > priority(\boldsymbol{\sigma_j})$. Then it is possible to use a recursive formulation [25] to compute the I-AUV quasi-velocities according to the established priorities:

$$\begin{aligned} i = 0: \quad \boldsymbol{\zeta_0} &= \mathbf{0}^n \\ \boldsymbol{P_0} &= \boldsymbol{I}^{n\times n} \\ i > 0: \quad \bar{\boldsymbol{J}}_i(\mathbf{q}) &= \boldsymbol{J}_i(\mathbf{q})\boldsymbol{P}_{i-1} \\ \boldsymbol{P_i} &= \boldsymbol{P}_{i-1} - \bar{\boldsymbol{J}}_i^{\dagger}(\mathbf{q})\bar{\boldsymbol{J}}_i(\mathbf{q}) \\ \boldsymbol{\zeta_i} &= \boldsymbol{\zeta}_{i-1} + \bar{\boldsymbol{J}}_i^{\dagger}(\mathbf{q})\,(\dot{\boldsymbol{\sigma}}_{id} - \boldsymbol{J}_i(\mathbf{q})\boldsymbol{\zeta}_{i-1}) \end{aligned} \quad (32)$$

## IV. COOPERATIVE UNDERWATER TRANSPORTATION

The goal of the cooperative underwater transportation is to perform a sequence of pick, transport, and place operations of a rigid object with a known geometry, using a group of floating I-AUVs. It is assumed that each of the robots has at least 6 DOF and can achieve any desired configuration of its EE (within the manipulator limits). Each robot has to be equipped with a navigation system, which is able to provide estimates of absolute vehicle position and orientation in the $N - frame$. The object grasping points are predefined and thus assumed to be known. To simplify the theoretical discussion, we assume that the cooperative transportation is performed using two robots transporting a pipe. The control strategy is to generate velocities in the $N - frame$, to be followed by the EEs of both robots, to move the pipe along a predefined path. Moreover, at each point of the path, a direction vector is defined reflecting the desired direction of the pipe axis. The aforementioned velocities are achieved by the robots separately by using a decentralized Task-Priority (TP) kinematic control algorithm. Although the whole system could be modeled and controlled using a centralized TP approach, that is not a practical solution. Running a control loop over underwater communications is not reliable, and it would require knowledge of the specifications of all robots, such as kinematic structure, joint limits, among others. The decentralized control permits each robot to run its own TP control, as when working independently. This embraces the modularity of the system and makes it easy to work with a non-homogeneous group of robots or even replace or reconfigure them if necessary. One of the robots is considered a master (**1**). It runs a state machine, controlling the sequence of operations, as well as the pipe path-following algorithm. It computes and communicates the desired EE velocity to the slave (**2**) and receives its EE configuration. It also sends simple commands to switch the mode of operation of the slave at certain moments of the sequence. Due to the low bandwidth of underwater acoustic communication channels, it is crucial to limit the amount of data that must be interchanged between the robots. Decentralized control permits running the sequencer and pipe setpoint controller at a lower rate than the internal robots' control, lowering the required bandwidth. Moreover, if the network fails, the robots' can continue with the last received setpoints, ensuring safety, and the mission control can recover if the network is briefly disrupted. Although it would be possible to use underwater Visual Light Communication (VLC) modems to overcome the bandwidth limitations, it would require careful planning of the modem placement as well as ensuring their line of sight, incurring additional costs. The control system scheme is presented in Fig. 4 and the following subsections describe the important blocks in detail.

### A. THE SEQUENCER

A state machine, running on the master vehicle, defines the current mode of operation and changes to a new one when required. Each mode enables a different subset of TP tasks from the dynamic task hierarchy $\mathcal{H} = \left\{T_{JL_1}, T_{NC_{\psi_2}}, T_{EEC_{\eta_{3.1}}} \mid T_{EEC_{v_{3.2}}}, T_{NC_{q_4}}\right\}$, where only one

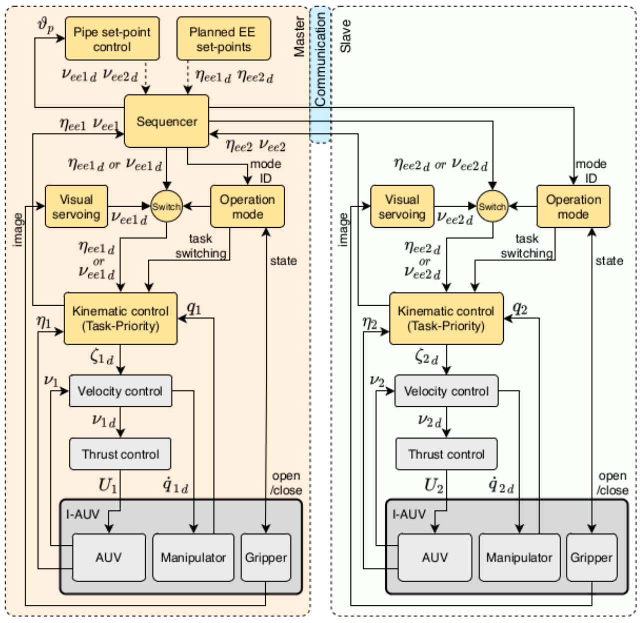


**FIGURE 4.** Cooperative transportation control scheme.

**TABLE 1.** Dynamic task priority hierarchy.

| Hierarchy | Type | Description | Approach | Pick | Transport | Place |
|---|---|---|---|---|---|---|
| Task 1 | $T_{JL}$ | Joint Limits | ✓ | ✓ | ✓ | ✓ |
| Task 2 | $T_{NC_\psi}$ | AUV yaw attitude | ✓ | ✓ | ✗ | ✓ |
| Task 3$_1$ | $T_{EEC_\eta}$ | EE Pose | ✓ | ✗ | ✗ | ✓ |
| Task 3$_2$ | $T_{EEC_\nu}$ | EE Velocity | ✗ | ✓ | ✓ | ✗ |
| Task 4 | $T_{NC_q}$ | Arm Configuration | ✓ | ✓ | ✗ | ✓ |

$T_{EEC}$ task is active at the time. Four modes of operation are defined (Table 1):

- **Approach**: used to approach the neighborhood of the pick-up points. This mode is triggered when both robots detect the pipe pose (see Sec. V-D), and both detections agree up to a margin error. Then, this mode is triggered, and each robot $R$ receives its pre-grasping EE configuration setpoint $\boldsymbol{\eta_{ee R_d}}$ to be accomplished (task $T_{EEC_\eta}$). The AUV yaw (task $T_{NC_\psi}$) is fixed normal to the direction of the pipe, to ensure visibility of the target. It also keeps both robots parallel to each other, reducing the risk of collision.
- **Pick**: used to perform grasping using visual feedback from the end-effector camera (see Sec. V-E). During this mode of operation, the end-effector is controlled in velocity mode (task $T_{EEC_\nu}$). Task $T_{NC_\psi}$ and $T_{NC_q}$ are maintained to keep a similar system configuration while approaching the pipe.
- **Transport**: intended for the cooperative transportation phase. As explained in Section IV, during the pipe transportation the robots operate in velocity mode to reduce the stress on the pipe and the manipulators (task $T_{EEC_\nu}$). While this mode of operation is enabled, the waypoints are continuously updated by the pipe setpoint controller. This is the only mode that requires regular communication, as the master sends the desired slave's EE velocity ($\boldsymbol{\nu_{ee2_d}}$) and this reports its current EE configuration and velocity ($\boldsymbol{\eta_{ee2}}$ $\boldsymbol{\nu_{ee2}}$).
- **Place**: this mode is triggered when the robots' EEs are located at the placement pose. Then, the EE maintains position and attitude ($T_{EEC_\eta}$) while the $T_{NC_\psi}$ task ensures AUV orthogonality with the pipe. At this point, the grippers open to drop the target in place.

In all modes of operation, the joint limits are secured by the $T_{JL}$ task, which is always enabled.

## B. PIPE PATH-FOLLOWING

The cooperative transportation occurs during the *transport mode*, when the path-following algorithm guides the pipe along a path defined as a sequence of waypoints $\boldsymbol{\vartheta_{p_i}} = \left[\boldsymbol{\eta_{p_1}^T}\ \boldsymbol{o_p^T}\right]^T$. The pipe position ($\boldsymbol{\eta_{p_1}}$) is the location of its origin ($\boldsymbol{P}$), defined as the point lying on the pipe axis which is equidistant to both holding points ($\boldsymbol{P_1}, \boldsymbol{P_2}$). During transport, the positions of the holding points should match those of the robots' EE ($\boldsymbol{P_1} = \boldsymbol{\eta_{ee2_1}}$ and $\boldsymbol{P_2} = \boldsymbol{\eta_{ee1_1}}$), and the pipe axis ($\boldsymbol{o_p}$) is given by the vector $\overrightarrow{\boldsymbol{P_1}\ \boldsymbol{P_2}}$. The control algorithm computes the linear ($\boldsymbol{\nu_p}$) and angular ($\boldsymbol{\omega_p}$) velocities that should be followed by the pipe in order to reach the target waypoint ($\boldsymbol{\vartheta_{p_i}}$). As shown in Fig. 5, the desired linear ($\boldsymbol{\nu_{p_d}}$) and angular velocities ($\boldsymbol{\omega_{p_d}}$) of the pipe can be chosen proportional to the position ($\boldsymbol{e_p}$) and orientation ($\boldsymbol{e_o}$) errors respectively:

$$\begin{aligned} \boldsymbol{e_p} &= \boldsymbol{P_d} - \boldsymbol{P}\ ;\ \boldsymbol{\nu_p} = K_p\boldsymbol{e_p} \\ \boldsymbol{e_o} &= \boldsymbol{o_p} \times \boldsymbol{o_{p_d}}\ ;\ \boldsymbol{\omega_p} = K_o\boldsymbol{e_o} \end{aligned} \tag{33}$$

Next, the desired linear velocities for $\boldsymbol{P_1}$ ($\boldsymbol{\nu_{1_d}}$) and $\boldsymbol{P_2}$ ($\boldsymbol{\nu_{2_d}}$) required to achieve the desired pipe velocity $[\boldsymbol{\nu_{p_d}^T}\ \boldsymbol{\omega_{p_d}^T}]^T$ are derived as follows:

$$\begin{aligned} \boldsymbol{\nu_{1_d}} &= \boldsymbol{\nu_{p_d}} + \boldsymbol{\nu_{o_1}}\ ;\ \boldsymbol{\nu_{o_1}} = \boldsymbol{\omega_{p_d}} \times (\boldsymbol{P_1} - \boldsymbol{P}) \\ \boldsymbol{\nu_{2_d}} &= \boldsymbol{\nu_{p_d}} + \boldsymbol{\nu_{o_2}}\ ;\ \boldsymbol{\nu_{o_2}} = \boldsymbol{\omega_{p_d}} \times (\boldsymbol{P_2} - \boldsymbol{P}) \end{aligned} \tag{34}$$

Finally, the desired EE velocity for each robot is computed:

$$\begin{aligned} \boldsymbol{\nu_{ee1_d}} &= \left[\boldsymbol{\nu_{1_d}^T}\ \boldsymbol{\omega_p^T}\right]^T \\ \boldsymbol{\nu_{ee2_d}} &= \left[\boldsymbol{\nu_{2_d}^T}\ \boldsymbol{\omega_p^T}\right]^T \end{aligned} \tag{35}$$

and fed to the corresponding $T_{EE_\nu}$ task running on both vehicles.

## C. DECENTRALIZED VELOCITY NORMALIZATION

It is worth noting that the resulting EE velocities ($\boldsymbol{\nu_{ee1_d}}, \boldsymbol{\nu_{ee2_d}}$) might not be achievable by the robots due, for instance, to the velocity limits of the joints. Therefore, the achieved EE velocities may differ from the desired ones, resulting in incorrect tracking of the desired pipe velocity.

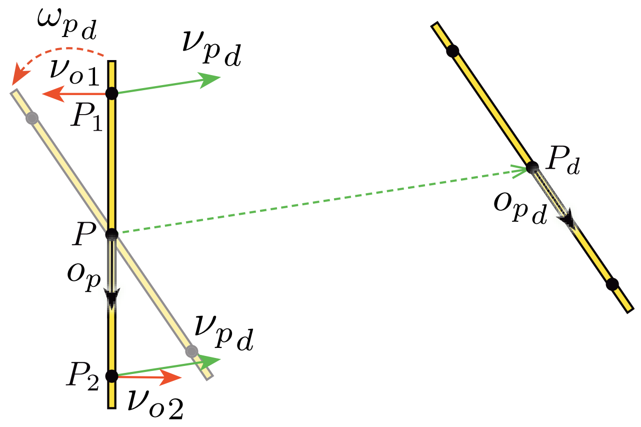


**FIGURE 5.** EE linear and angular velocity setpoints to reach a pipe waypoint.

In order to keep the system decentralized, a novel method is proposed to scale the desired EE velocity of both I-AUVs (Alg. 1). The method uses an inverse scaling factor $s$, which is iteratively adapted. First, both velocity setpoints computed by the task priority hierarchy of each robot are scaled according to the inverse scale factor of the previous iteration ($s_{k-1}^{-1}\boldsymbol{\nu_{ee_1,d}}$, $s_{k-1}^{-1}\boldsymbol{\nu_{ee_2,d}}$). Then, their discrepancies with the previously achieved end-effector velocities ($\boldsymbol{\Delta\nu_{ee_1}}$ and $\boldsymbol{\Delta\nu_{ee_2}}$) are computed (lines 1 and 2). If the discrepancies ($\Delta\nu_{ee_j}(i)$) of all the Cartesian components ($i \in \{1..6\}$) of both setpoints ($j \in \{1, 2\}$) are all smaller than a threshold ($\varepsilon$), then the inverse scaling factor is lowered (line 4), but never allowed to fall below 1. This ensures that the inverse scaling factor always scales down the requested velocities and never amplifies them. Otherwise, if any component discrepancy overpasses the threshold, the maximum discrepancy observed ($m$) in any of the ($i$) components of both setpoints ($j$) is computed (line 8), being used to increase the absolute value of the inverse scale $s$ by a maximum increment of $\epsilon$ (line 9). Increasing $s$ will tend to decrease the discrepancies in the next iteration. Finally, the new factor $s$ is smoothed through an exponential filter (line 11) before applying it to scale both velocity setpoints (lines 12 and 13).

**Algorithm 1:** Adaptive Scaling of the Desired EEs Velocity

```
1  $\boldsymbol{\Delta\nu_{ee1}} = s_{k-1}^{-1}\boldsymbol{\nu_{ee1_d}} - \boldsymbol{\nu_{ee1,k-1}}$
2  $\boldsymbol{\Delta\nu_{ee2}} = s_{k-1}^{-1}\boldsymbol{\nu_{ee2_d}} - \boldsymbol{\nu_{ee2,k-1}}$
3  if $\forall j, i\ \Delta\nu_{eej}(i) < \varepsilon$ then
4      $s_k = \texttt{Max}\,(0.95 \cdot s_{k-1}, 1.0)$
5  else
6      $\boldsymbol{\Delta\nu_{ee1}} = \boldsymbol{\nu_{ee1_d}} - \boldsymbol{\nu_{ee1,k-1}}$
7      $\boldsymbol{\Delta\nu_{ee2}} = \boldsymbol{\nu_{ee2_d}} - \boldsymbol{\nu_{ee2,k-1}}$
8      $m = \texttt{Max}\,(\forall j, i\ \Delta\nu_{eej}(i) < \varepsilon)$
9      $s_k = \begin{cases} s_{k-1} + \epsilon & \text{if } k_\epsilon(m - s_{k-1}) > \epsilon, \\ s_{k-1} - \epsilon & \text{if } k_\epsilon(m - s_{k-1}) < -\epsilon, \\ s_{k-1} + k_\epsilon(m - s_{k-1}) & \text{otherwise.} \end{cases}$
10 end
11 $s_k = \alpha s_k + (1 - \alpha)s_{k-1}$
12 $\boldsymbol{\nu'_{ee1_d}} = \boldsymbol{\nu_{ee1_d}} \cdot s_k^{-1}$
13 $\boldsymbol{\nu'_{ee2_d}} = \boldsymbol{\nu_{ee2_d}} \cdot s_k^{-1}$
```

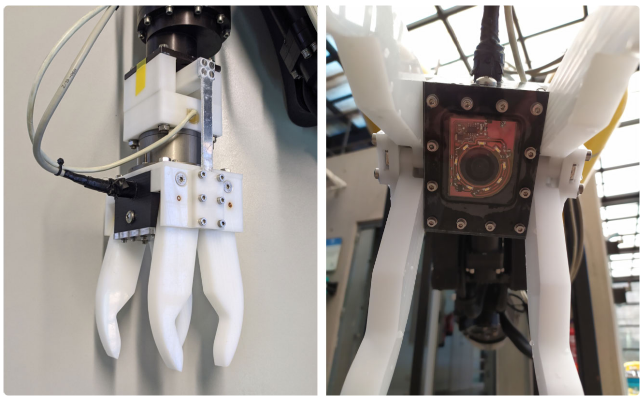


**FIGURE 6.** Customized end-effector with embedded camera.

## V. IMPLEMENTATION

### A. PLATFORMS

The team of robots consists of two GIRONA500 lightweight, modular I-AUVs designed and developed at the University of Girona [26], which can be reconfigured for different tasks by changing their payload and thruster configuration. Each vehicle was used in a 4 DOF configuration (yaw, surge, sway and heave), being passively stable in pitch and roll by design. The manipulators employed are an ECA 5E Micro and an ECA 5E Mini, both with four rotational joints actuated by electric screw drives, which makes them powerful but slow. Nevertheless, the drives exhibit high friction forces resulting in a velocity dead-band zone. The kinematic structure of both UVMSs is presented in Fig. 1. In order to correctly grasp the pipe, a custom end-effector has been designed and built (see Fig. 6). The shape of the fingers helps to drive the pipe to the end-effector center.

Additionally, three cameras are being used for each robot: 1) An analog camera attached to the body of the GIRONA500, which is forward-looking, and is used to localize the pipe based on the detection of ARUCO markers [27]; 2) An analog camera embedded in the end-effector, used to provide visual feedback to improve the grasping accuracy, and 3) A high-performance megapixel down-looking industrial camera, attached to the lower side of the I-AUV, used for getting seafloor image updates for the navigation filter.

### B. SOFTWARE ARCHITECTURE

GIRONA500 uses the component-oriented layer-based software architecture for autonomy (COLA2) [28], integrating perception, navigation, guidance, and control. The navigation layer implements an extended Kalman filter (EKF) for sensor fusion to estimate the robot's position and velocity. The AUV control is based on a nested Task Priority/velocity PID controller. The velocity controller computes the force and torque ($\tau_d$) to be applied to reach the desired velocity ($v_d$). This value is computed by combining a standard 4-DOF PID control with a feed-forward model, providing the nominal force to be applied to achieve a certain velocity. The input to the velocity controller ($\tau_d$) is allocated to the thrusters using the thruster allocation matrix within the thruster control block. Once the force to be applied by each thruster is known, a static thruster model is used to convert the force into thruster setpoints. The ECA manipulators can be controlled either in voltage (open loop) or in velocity, based on an embedded PID controller. However, this control is applied to the electric screw drives instead of the manipulator's joints. Moreover, the manipulator provides position feedback from digital Hall sensors, which requires the arm to be calibrated when powered on. As a result, an extra layer has been added to the COLA2 architecture to elevate the control and feedback to the joint level and to handle the calibration procedure. The calibration procedure is performed forcing the actuator limits, which adds a small uncertainty as the limits are not always triggered precisely at the same value.

The distributed framework for cooperative pipe transportation is constituted by three main blocks:

- The Mission Manager runs the mission sequencer, a state machine that controls the sequence of operations of the robots and generates setpoints for the robots' EEs, either in configuration or velocity, depending on the sequence stage.
- The Action Layer defines a set of modes of operation that encode the prioritized tasks to be executed by the Kinematic Control Layer. These modes of operation are detailed below in Table 1. In addition, this layer runs vision-based algorithms used to compute the pipe pick and place configurations as well as to guide the EE during the grasping stage.
- The Kinematic Control Layer implements the TP control framework and is in charge of accomplishing the control objectives in a reactive fashion. The theoretical formulation of the algorithm is presented in Section III. However, the actual implementation requires multiple extensions and practical solutions, which were described in detail in previous works by the authors [8], [11].

One robot is assigned the master role, which is in charge of executing the Mission Sequencer. The COLA2 architecture, the Action Layer, and the Kinematic Control Layer are executed independently on each robot.

## C. ROBOT LOCALIZATION

The vehicle relies on a dead-reckoning estimate to navigate, based on an Extended Kalman Filter (EKF), which is in charge of integrating the information from different sensors to estimate the robot's position and velocity ($\boldsymbol{x_k} = [\boldsymbol{\eta}_1^T \boldsymbol{v}_1^T]^T$). The prediction stage relies on a simple constant velocity kinematics model with attitude input from the Attitude and Heading Reference System (AHRS) to estimate how the state evolves from time $k-1$ to time $k$. Lineal velocity readings from the Doppler Velocity Log (DVL) are used to provide updates to the filter. Moreover, in order to bound the inherent localization drift and improve pose estimation, absolute measurements (e.g., GPS, USBL) can be used to update the navigation filter. However, neither GPS nor USBL fixes are possible within a water tank. Therefore, the DVL-AHRS based dead-reckoning positioning is not accurate enough for manipulation tasks, where centimeter positioning accuracy is required. This is even worse in cooperative tasks where distributed accumulated errors can lead to unpredictable behavior. Therefore a vision-based localization method using an *a priori* map of the environment has been introduced. When the system operates in the water tank, a poster image (Fig. 7) is placed on the bottom and a vision-based navigation algorithm processes the images gathered with the down-looking camera. Extracting features from them and solving the image-to-poster registration allows computing the AUV pose to provide absolute navigation updates for the filter. An article detailing this method is being prepared for publication.

## D. PIPE LOCALIZATION

Estimation of the pick and place locations is achieved by observing ArUco markers [27], which represents a good trade-off between accuracy and performance. In our experiments, the pipe lies on a pair of v-shaped hangers mounted on aluminum stands, with ArUco markers placed below the hangers to estimate the holding points ($\boldsymbol{H_1}, \boldsymbol{H_2}$). The pipe center $\boldsymbol{P}$ is considered equidistant to both holding points, and the direction $\boldsymbol{r}$ is aligned with $\overrightarrow{\boldsymbol{H_1 H_2}}$. A predefined distance $d_1$ is then used to approximate the grasping points ($\boldsymbol{P_1}, \boldsymbol{P_2}$):

$$\begin{aligned} \boldsymbol{P} &= \frac{1}{2}(\boldsymbol{H_2} - \boldsymbol{H_1}) \\ \boldsymbol{r} &= \frac{(\boldsymbol{H_2} - \boldsymbol{H_1})}{|\boldsymbol{H_2} - \boldsymbol{H_1}|} \\ \boldsymbol{P_1} &= \boldsymbol{P} - \boldsymbol{r} \cdot d_1 \\ \boldsymbol{P_2} &= \boldsymbol{P} + \boldsymbol{r} \cdot d_1 \end{aligned} \tag{36}$$

The pre-grasping EEs configuration ($\boldsymbol{\eta_{ee1}}, \boldsymbol{\eta_{ee2}}$) is predefined in the robot's manipulation parameters, where $d_2$ is the pre-grasping distance and $\alpha$ the orientation angle (see Fig. 8). This configuration is defined with respect to the detected pipe position $\boldsymbol{P}$ and orientation vector $\boldsymbol{r}$ in the $N-frame$.

Let the orientation of the pipe in the $N-frame$ be represented by a $3 \times 3$ matrix $\boldsymbol{R_p} = [\boldsymbol{r_{p_1}}\, \boldsymbol{r_{p_2}}\, \boldsymbol{r_{p_3}}]$ where $\boldsymbol{r_{p_1}} = (-\boldsymbol{r}) \times [0\,0\,1]^T, \boldsymbol{r_{p_2}} = -\boldsymbol{r}$ and $\boldsymbol{r_{p_3}} = \boldsymbol{r_{p_1}} \times \boldsymbol{r_{p_2}}$. Then the desired grasping orientation can be computed by simply rotating the $R_p$ frame through an angle of $\alpha$ around the $y$ axis:

$$\boldsymbol{R_d} = \boldsymbol{R_y(\alpha)} \cdot \boldsymbol{R_p} = [\boldsymbol{r_{d_1}}\, \boldsymbol{r_{d_2}}\, \boldsymbol{r_{d_3}}]^T \tag{37}$$

Finally, each pre-grasping pose is given by:

$$\begin{aligned} \boldsymbol{\eta_{1ee_1}} &= \boldsymbol{P_1} - d_2 \boldsymbol{r_{d_3}} \\ \boldsymbol{\eta_{1ee_2}} &= RPY2Euler(\boldsymbol{R_d}) \\ \boldsymbol{\eta_{2ee_1}} &= \boldsymbol{P_2} - d_2 \boldsymbol{r_{d_3}} \\ \boldsymbol{\eta_{2ee_2}} &= RPY2Euler(\boldsymbol{R_d}) \end{aligned} \tag{38}$$

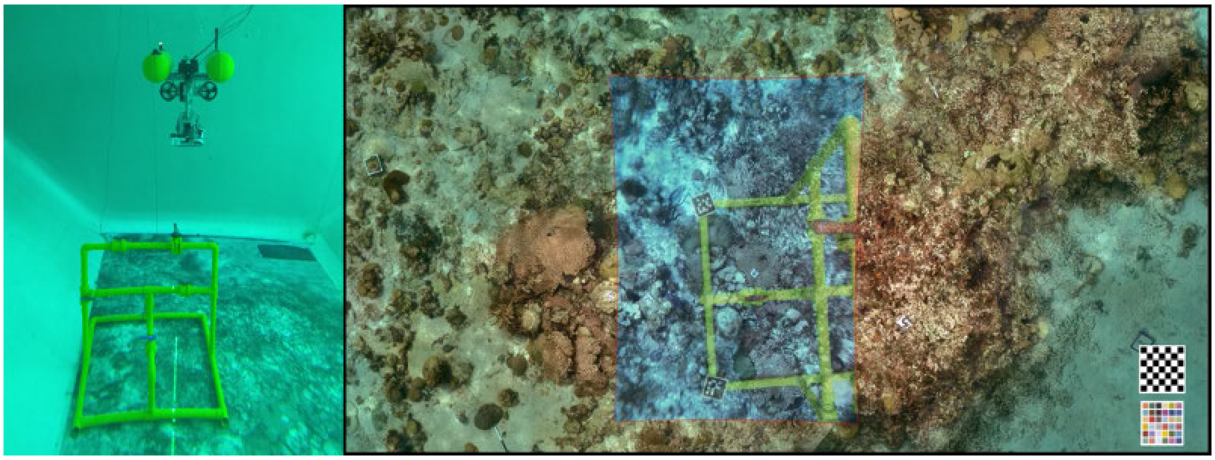


**FIGURE 7.** Left: Water tank scenario with the poster placed on the bottom. Right: Example of a camera image registered to the known poster illustrating the achieved accuracy.

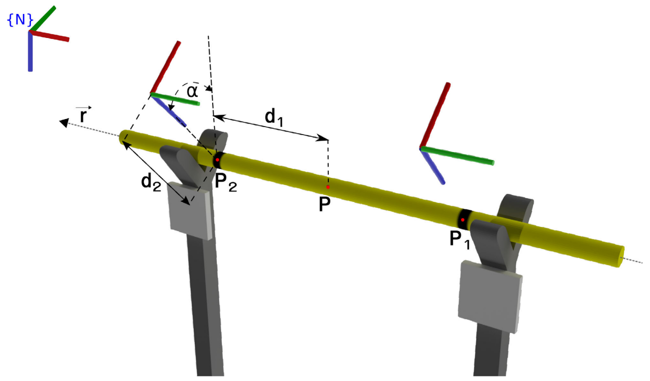


**FIGURE 8.** Pre-grasping EE configuration setpoints. ***P*** and ***r*** are calculated based on the stand markers detection, whereas $d_1$, $d_2$, and $\alpha$ are user-defined.

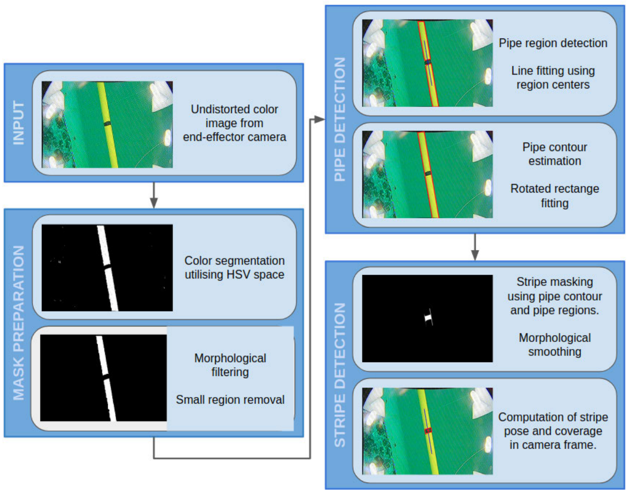

INPUT
Undistorted color image from end-effector camera
MASK PREPARATION
Color segmentation utilising HSV space
Morphological filtering
Small region removal
PIPE DETECTION
Pipe region detection
Line fitting using region centers
Pipe contour estimation
Rotated rectange fitting
STRIPE DETECTION
Stripe masking using pipe contour and pipe regions.
Morphological smoothing
Computation of stripe pose and coverage in camera frame.

**FIGURE 9.** Stripe detection process.

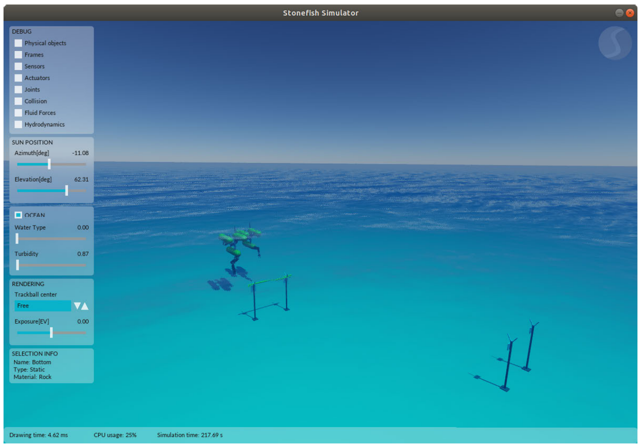
where the function $RPY2Euler(\cdot)$ computes the Euler angles of the input rotation matrix.

### *E. VISUAL SERVOING*

Using visual feedback to control the end-effector movement is a common approach to achieve a robust execution of the grasping in real-world scenarios, mainly due to the difficulty of locating the end-effector with enough accuracy in the $N - frame$. Black stripes have been placed over the yellow-colored pipe to mark the grasping points ($\boldsymbol{P}1, \boldsymbol{P}2$). The algorithm steps, shown in Fig. 9, follow: First, the input image is converted to the HSV (hue-saturation-value) space to be ready for segmentation. HSV space is better for color segmentation as it separates color information from intensity or lighting. The segmented image is used to create a binary mask that can be applied to the original image, extracting only the colored pipe related pixels. This mask is then used to fit a line to the pipe body and estimate the entire pipe contour. Knowing the pipe contour and the colored pipe regions, the stripe region can be estimated. Finally, the stripe center is used to compute the end-effector alignment error, and the stripe area is used to estimate the distance to the pipe and set the EE desired forward velocity to approach the picking point. If the detection of the stripe fails, the previous EE velocity setpoint is kept, up to a maximum number of consecutive failed detections, after which the visual servoing would abort.

## VI. SIMULATION

The core simulation tool used in this work was the open-source *Stonefish* C++ simulation library [29] combined with the Robot Operating System (ROS) interface package called *stonefish_ros*. This software is specifically designed for the simulation of marine robots. It delivers full support for rigid body dynamics of kinematic trees (like I-AUVs), geometry-based hydrodynamics, buoyancy and collision detection. It also simulates all underwater sensors and actuators to seamlessly replace the real system with a simulated robot. The full COLA2 software architectures of both robots run simultaneously with the simulator to perfectly mimic real experiments.



Azimuth[deg] -11.08
Elevation[deg] 62.31
Water Type 0.00
Turbidity 0.87
Exposure[EV] 0.00
Name: Bottom
Type: Static
Material: Rock
Drawing time: 4.62 ms CPU usage: 25% Simulation time: 217.69 s

**FIGURE 10.** Simulated scenario using Stonefish.

The simulation setup for the cooperative pipe transportation is composed of the full model of two cooperating GIRONA500 I-AUV, each one equipped with its specific manipulator (described in Section V-A) and sensor suite, a flat sea bottom, two pairs of pipe stands and the pipe to be transported. The pipe stands are equipped with ARUCO markers [27] and the pipe appearance reflects the real one, enabling pipe detection and visual servoing using the virtual images (see Section V-D and Section V-E). Thanks to collision detection and the use of the real geometry of the robots' grippers and pipe, the grasping and transportation are precisely simulated, allowing for the identification of problems before water tank trials. Unique features of *Stonefish*, such as the simulation of a wrist-mounted force-torque sensor, allow for in-depth analysis of the system's performance and fast development of practical solutions.

## A. SCENARIO

In the simulated scenario, two pairs of pipe stands ($\boldsymbol{S_{12}}, \boldsymbol{S_{34}}$) lie on the sea bottom at five meters depth, each one separated by 1.5 meters from its pair. The pose of the stands with respect to the $N - frame$ $(x, y, \theta)$ is given by: $\boldsymbol{S_{12}} = [0, 0.75, 0]^T$, $\boldsymbol{S_{34}} = [8, -3.25, 0]^T$. Since the pipe is kept parallel to the XY plane, pipe configurations are defined as $\boldsymbol{\vartheta} = [x, y, z, \varphi]^T$. The initial pipe configuration is $\boldsymbol{\vartheta} = [0, 0.75, 3.5, 0]^T$, laying on top of $\boldsymbol{S_{12}}$. Once the pipe is detected, each robot pre-grasping EE configuration ($\boldsymbol{\eta_{ee1}}, \boldsymbol{\eta_{ee2}}$) is computed and the robots are driven to it sequentially. Then, both robots simultaneously approach the grasping points ($\boldsymbol{P_1}, \boldsymbol{P_2}$) using visual guidance to finally grasp the pipe. After the pipe is fully grasped, the current pipe configuration $\boldsymbol{\vartheta_0}$ is defined, being assumed to be equidistant to both pipe holding points ($\boldsymbol{P_1}, \boldsymbol{P_2}$) and aligned with $\overrightarrow{\boldsymbol{P_1}\,\boldsymbol{P_2}}$. Next, the robot is guided along six waypoints defined with respect to $\boldsymbol{\vartheta_0}$: $\boldsymbol{\vartheta_1} = [0, 0, -0.5, 0]^T$, $\boldsymbol{\vartheta_2} = [4, 0, -0.5, 0]^T$, $\boldsymbol{\vartheta_3} = [4, 0, -0.5, \pi/2]^T$, $\boldsymbol{\vartheta_4} = [4, -4, -0.5, \pi/2]^T$, $\boldsymbol{\vartheta_5} = [4, -4, -0.5, 0]^T$ and $\boldsymbol{\vartheta_6} = [8, -4, -0.5, 0]^T$. The placement waypoint $\boldsymbol{\vartheta_7}$ is computed once the second pair of stands are detected.

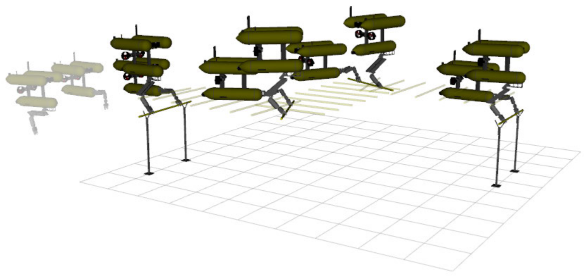


**FIGURE 11.** The UVMS team performing the pick, transport, and place operations on simulation. Five states out of the performed trajectory are shown.

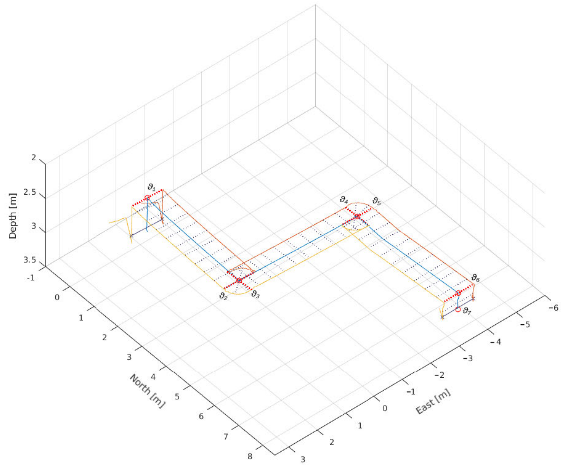


**FIGURE 12.** Pipe trajectory (blue) and robots' EE trajectories (yellow, orange) during cooperative transportation in the $N - frame$. Pipe orientation has been sampled over the trajectory (black dotted lines). Pipe waypoints are marked in red. Initial and final pipe configurations are shown as solid black lines.

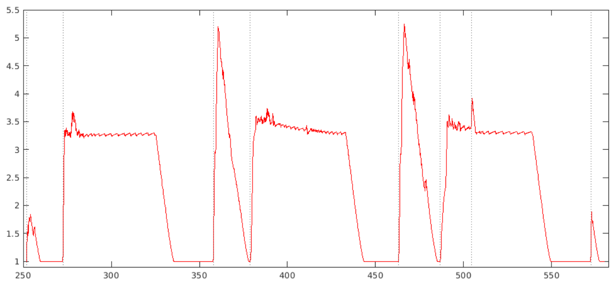


**FIGURE 13.** Evolution of scaling factor according to Alg. 1.

## B. RESULTS

Fig. 12 presents the trajectory followed by the pipe and both EEs, demonstrating how the UVMS team successfully grasped the pipe from $\boldsymbol{S_{12}}$ stand, followed the predefined sequence of pipe waypoints, and placed it on the $\boldsymbol{S_{34}}$ stand. In addition, Fig. 11 shows the executed mission sampled over time to provide greater insight.

Fig. 14 reports the evolution of the EE's velocity in the $N - frame$ during pipe transportation, whereas Fig. 13 presents the evolution of the scaling factor according to Alg. 1. It can be

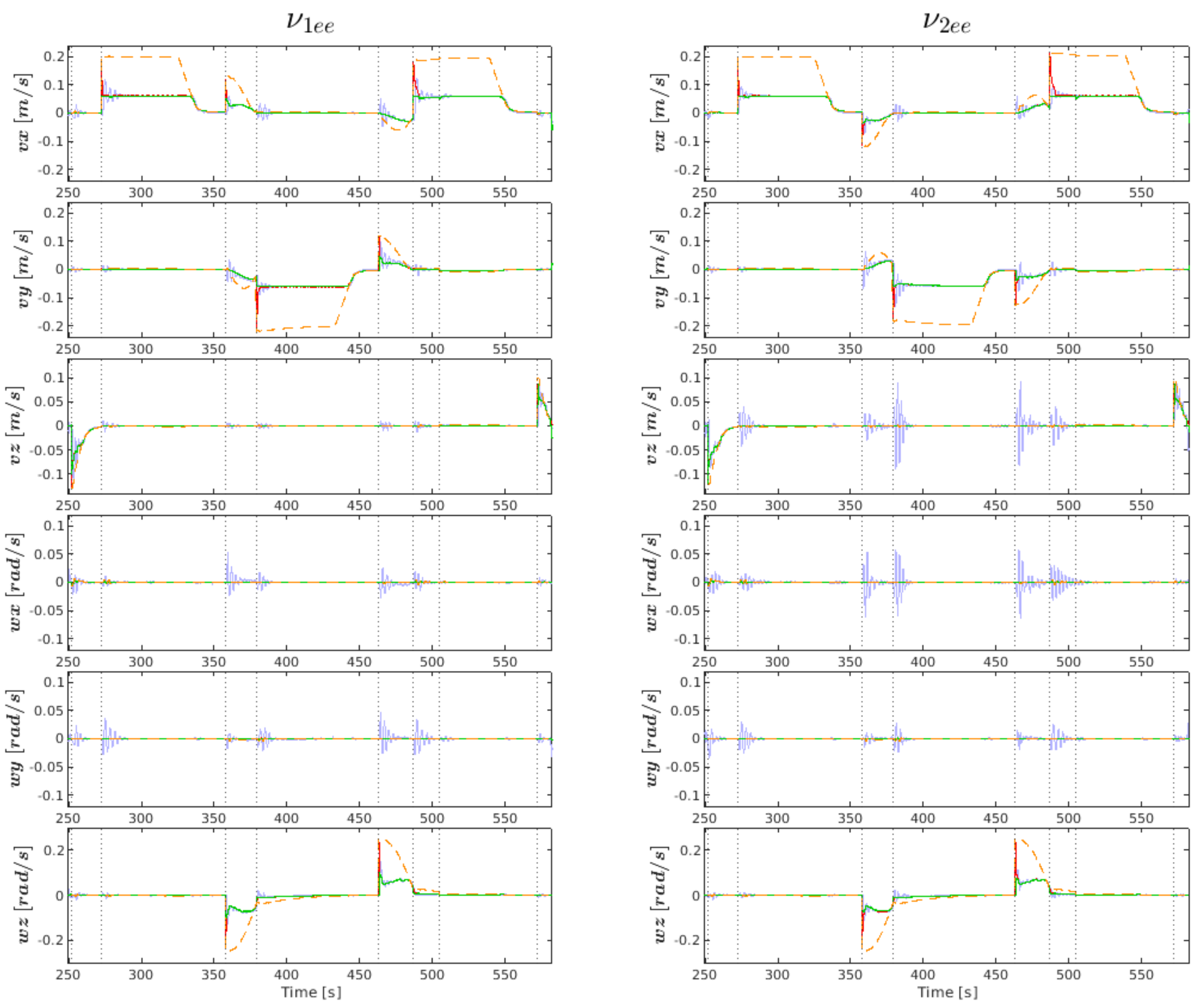


**FIGURE 14.** EE's velocity in the $N-frame$ during pipe transportation. The plot includes the unscaled setpoints (orange), the adapted setpoints (red), the internal controllers' velocity requests (green) and the internal controllers' feedback (blue). Setpoint changes are marked by vertical dotted lines.

seen that the velocity setpoints rapidly adapt to the robots' limitations.

A video demonstrating the whole mission using the Stonefish simulator can be found at the following url: https://youtu.be/iBnxSGs2t1U.

## VII. EXPERIMENTAL VALIDATION

For the experimental validation, two GIRONA500 I-AUVs were used, one belonging to the University of Girona (the original prototype [26]) and another belonging to the University Jaume I and built by Iqua Robotics S.L (http://iquarobotics.com/). Each one was equipped with a 4 DOF ECA electric manipulator (see section V-A). The experiment was performed in the water tank of the University of Girona, Underwater Robotics Lab (CIRS). Thanks to the transparency of the Stonefish simulator, the same software architecture was used during the simulation and experimental

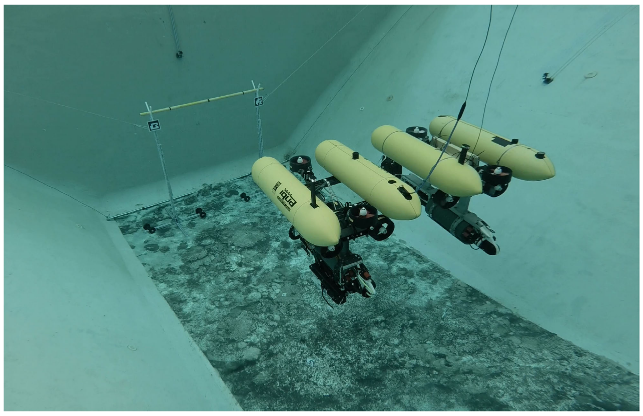
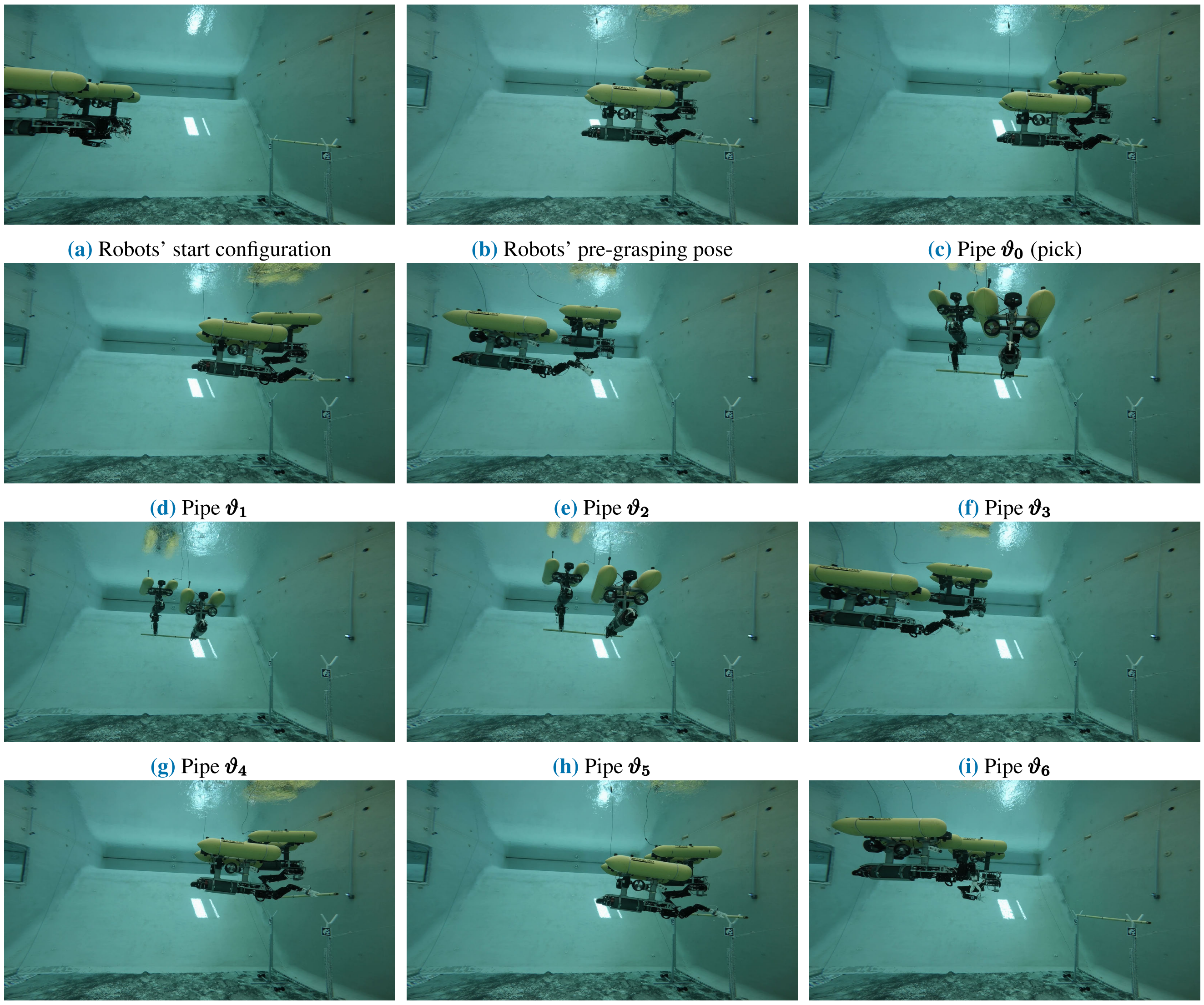


**FIGURE 15.** Water tank setup.

trials. One additional component was used during the experiments - the vision-based navigation, reported in section V-C.



(a) Robots' start configuration (b) Robots' pre-grasping pose (c) Pipe $\boldsymbol{\vartheta}_0$ (pick)

(d) Pipe $\boldsymbol{\vartheta}_1$ (e) Pipe $\boldsymbol{\vartheta}_2$ (f) Pipe $\boldsymbol{\vartheta}_3$

(g) Pipe $\boldsymbol{\vartheta}_4$ (h) Pipe $\boldsymbol{\vartheta}_5$ (i) Pipe $\boldsymbol{\vartheta}_6$

(j) Pipe $\boldsymbol{\vartheta}_7$ (k) Pipe $\boldsymbol{\vartheta}_8$ (place) (l) Robots' retreat pose

**FIGURE 16. Mission sequence in the water tank.**

Although this component can also be simulated in Stonefish, it would require a lot of computational resources which can be avoided by simply lowering the simulated noise in the navigation sensors.

### A. SCENARIO

The scenario, similar to the one used during the simulation, was then adapted to the dimensions of the water tank. The pipe was lying on a pair of stands separated by roughly 1.6 meters, having the grasping points marked with black stripes, 1.2 meters apart. The mission consisted of picking up the pipe, following a predefined L-shaped pipe trajectory, and placing it back to its original place. Similarly to the simulated scenario, the robots were sequentially commanded to a pre-grasping EE configuration after locating the pipe stands. Then, simultaneously, they approached the grasping points using visual servoing and grasped the pipe. Next, eight waypoints define the desired pipe trajectory, with respect to the initial pipe configuration $\boldsymbol{\vartheta}_0$: $\boldsymbol{\vartheta}_1 = [0, 0, -0.5, 0]^T$, $\boldsymbol{\vartheta}_2 = [-2.5, 0, -0.5, 0]^T$, $\boldsymbol{\vartheta}_3 = [-2.5, 0, -0.5, \pi/2]^T$, $\boldsymbol{\vartheta}_4 = [-2.5, 2.2, -0.5, \pi/2]^T$, $\boldsymbol{\vartheta}_5 = [-2.5, 0.6, -0.5, \pi/2]^T$, $\boldsymbol{\vartheta}_6 = [-2.5, 0, -0.5, 0]^T$, $\boldsymbol{\vartheta}_7 = [0, 0, -0.5, 0]^T$, and $\boldsymbol{\vartheta}_8 = \boldsymbol{\vartheta}_0$. The transition from waypoints $\boldsymbol{\vartheta}_2$ to $\boldsymbol{\vartheta}_3$ defines a 90° rotation from the center of the pipe, whereas the transition from $\boldsymbol{\vartheta}_5$ to $\boldsymbol{\vartheta}_6$ defines a −90° rotation pivoting over the left robot's end-effector.

### B. RESULTS

Fig. 16 presents snapshots of the performed mission in the water tank at consecutive time instants. Fig. 17 shows the trajectory for the computed pipe center and both EEs during the transportation phase. This plot has been separated into two consecutive moments for better insight.

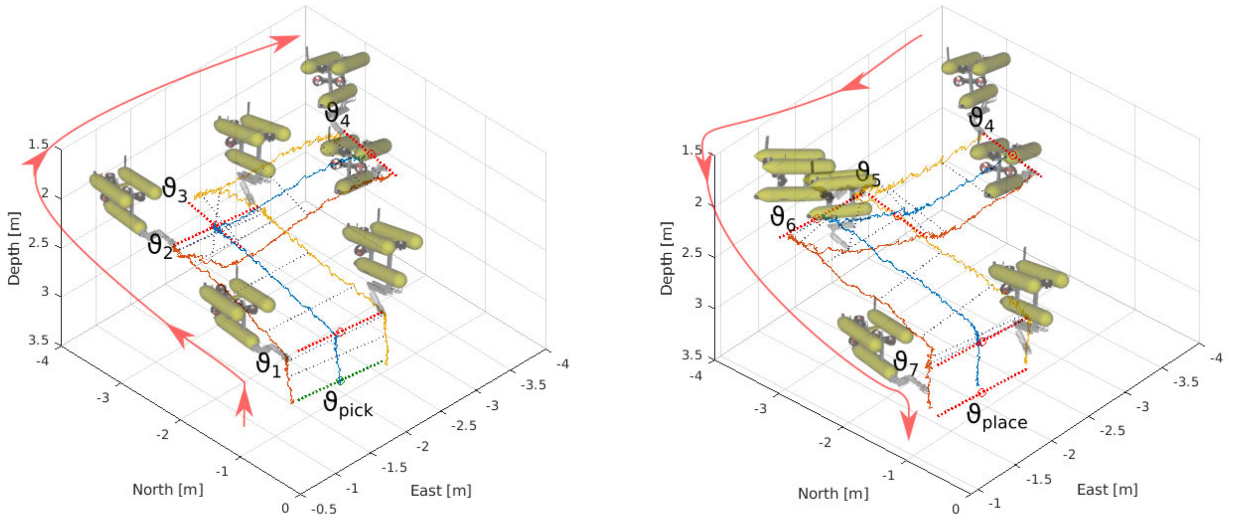


**FIGURE 17.** Back and forward pipe (blue) and robots' EE trajectories (yellow, orange) during the cooperative transportation phase. Pipe setpoints are marked in red.

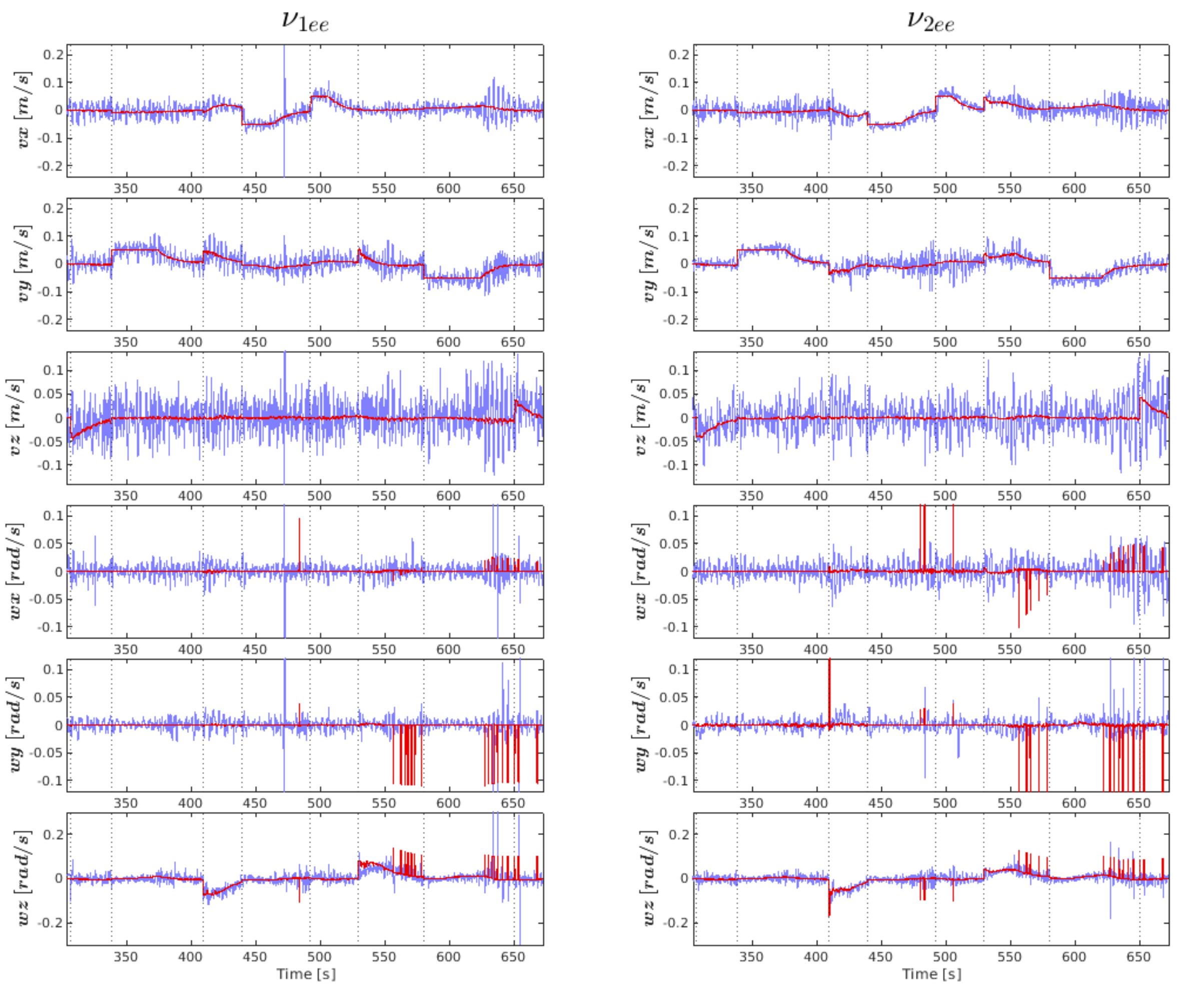


**FIGURE 18.** EEs velocity setpoints (red) and feedback (blue) in the $N-frame$ during pipe transportation. Setpoint changes are marked by vertical dotted lines.

The robots can be seen to successfully perform a cooperative pipe transportation mission involving picking up the pipe, guiding it along the path through a sequence of waypoints, and placing it at its original site.

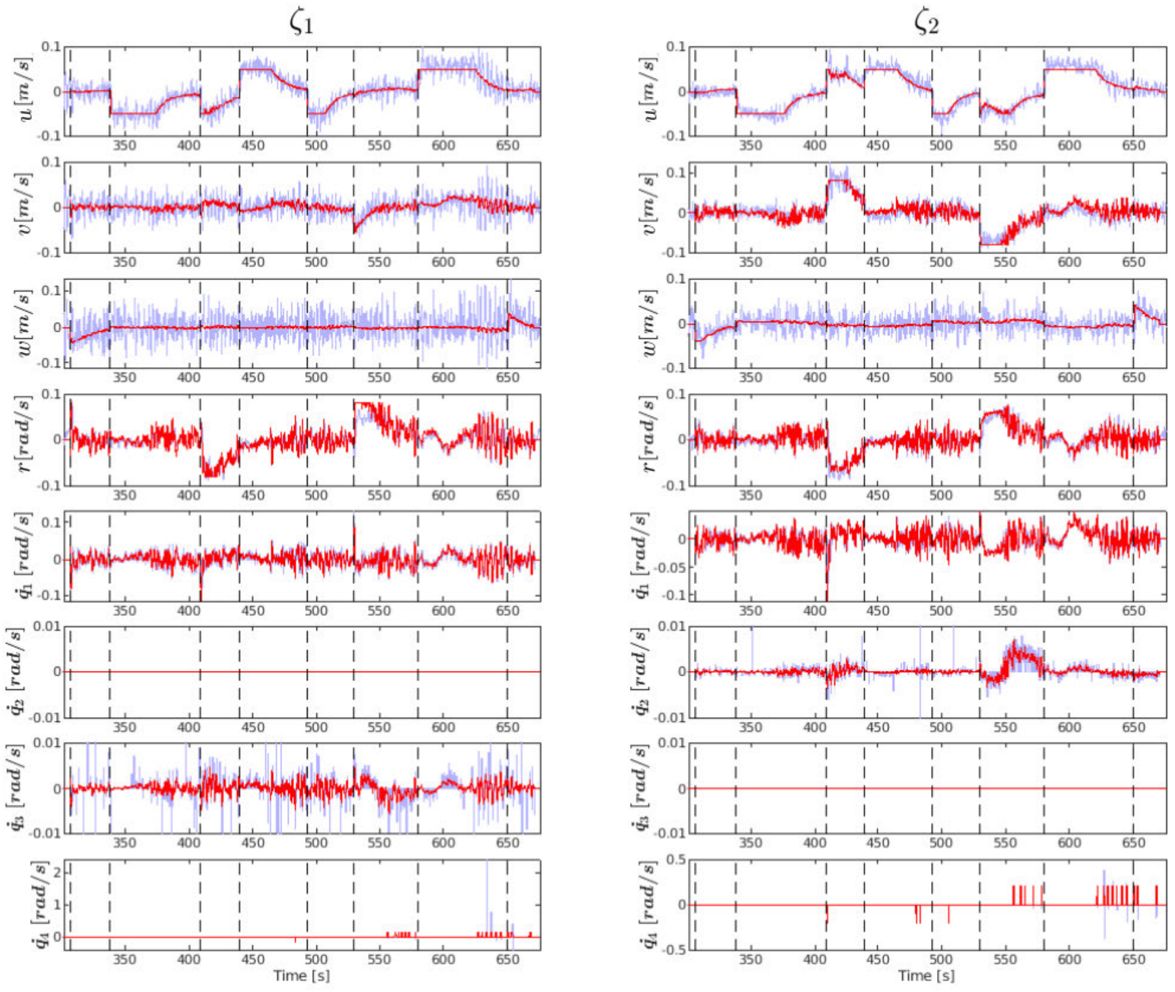


**FIGURE 19.** AUV and joint velocity setpoints (red) and feedback (blue) during pipe transportation. Setpoint changes are marked by vertical dotted lines.

Fig. 18 presents the control response (blue) of the end-effectors' requested velocities (red), showing how the robots were able to follow the EEs setpoints and Fig. 19 presents the actual UVMSs' quasi-velocities (vehicle and manipulator). It is worth noting that the manipulators' drives cannot achieve very-low speeds due to high internal friction forces. A custom TP extension is enabled to overcome this problem, which fixes the minimum achievable speed for those joints after the scaling, to avoid blocking the manipulator. Since the manipulator jaw suffers most from this problem, it is useful to note that it can be clearly seen that the generated spikes for $\dot{q}_4$ are aligned with the spikes visible in the EEs velocity requests.

In this experiment, due to the difficulty of operating acoustic modems in a small water tank, the communication between the robots was performed via an Ethernet cable, consistently limiting the bandwidth to simulate the real-world case. The cooperative transportation phase is the only one that requires continuous data transfer. On each iteration, the master sends an EE velocity request (6 DOF) to the slave, and this returns feedback of its EE position (6 DOF) and the achievable EE velocity (6 DOF) (see Sec. IV). For this experiment, standard ROS communications are used, and the cooperative control loop operates at 10 Hz. Without considering TCP/IP overhead, each message requires 28 bytes (6 float numbers $\times$ 4 bytes each float + 4 bytes for a timestamp), consuming a total of 6.72 kbps (3 messages $\times$ 28 bytes message $\times$ 10 Hz $\times$ 8 bits).

During the rest of the mission (approach, pick, place, and retreat), communication between the master and slave robots is only necessary for changing the mode of operation and its goal and send back an acknowledgment when it is successfully completed.

A video demonstrating the whole mission in the water tank can be found at the following url: https://youtu.be/epnU4v3Hz44.

## VIII. CONCLUSION

This paper has presented a decentralized cooperative manipulation and transportation scheme for a team of UVMSs.

Simulated and water tank results have demonstrated a fully autonomous cooperative transportation mission, including detection, pickup, transport, and placement operations of a bulky rigid object. The method is based on a decentralized kinematic control of two I-AUVs, simultaneously controlling their end-effector velocities to achieve a common goal. The decentralized control provides modularity and robustness to the system and enables operation under low bandwidth communication. The desired task priority hierarchies are grouped into modes of operation to allow changing the system's entire behaviour with simple messages. The simulated and experimental results validate the proposed method.

## REFERENCES

[1] H. H. Wang, S. M. Rock, and M. J. Lees, "Experiments in automatic retrieval of underwater objects with an AUV," in *Proc. Challenges Our Changing Global Environ. Conf. (OCEANS, MTS/IEEE)*, vol. 1, Oct. 1995, pp. 366–373.

[2] S. K. Choi, G. Y. Takashige, and J. Yuh, "Experimental study on an underwater robotic vehicle: ODIN," in *Proc. IEEE Symp. Auton. Underwater Vehicle Technol.*, Dec. 1994, pp. 79–84.

[3] V. Rigaud, E. Coste-Maniere, M. J. Aldon, P. Probert, M. Perrier, P. Rives, D. Simon, D. Lang, J. Kiener, A. Casal, J. Amar, P. Dauchez, and M. Chantler, "UNION: Underwater intelligent operation and navigation," *IEEE Robot. Autom. Mag.*, vol. 5, no. 1, pp. 25–35, Mar. 1998.

[4] D. M. Lane, J. B. C. Davies, G. Casalino, G. Bartolini, G. Cannata, G. Veruggio, M. Canals, C. Smith, D. J. O'Brien, M. Pickett, G. Robinson, D. Jones, E. Scott, A. Ferrara, D. Angelleti, M. Coccoli, R. Bono, P. Virgili, R. Pallas, and E. Gracia, "AMADEUS: Advanced manipulation for deep underwater sampling," *IEEE Robot. Autom. Mag.*, vol. 4, no. 4, pp. 34–45, Dec. 1997.

[5] J. Evans, P. Redmond, C. Plakas, K. Hamilton, and D. Lane, "Autonomous docking for intervention-AUVs using sonar and video-based real-time 3D pose estimation," in *Proc. OCEANS*, 2003, pp. 2201–2210.

[6] N. Palomeras, P. Ridao, D. Ribas, and G. Vallicrosa, "Autonomous I-AUV docking for fixed-base manipulation," *IFAC Proc. Volumes*, vol. 47, no. 3, pp. 12160–12165, Jan. 2014.

[7] A. Carrera, N. Palomeras, N. Hurtos, P. Kormushev, and M. Carreras, "Learning by demonstration applied to underwater intervention," in *Proc. 17th Int. Conf. Catalan Assoc. Artif. Intell. (CCIA)*, 2014, pp. 95–104.

[8] P. Cieslak, P. Ridao, and M. Giergiel, "Autonomous underwater panel operation by GIRONA500 UVMS: A practical approach to autonomous underwater manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2015, pp. 529–536.

[9] P. Cieślak, R. Simoni, P. R. Rodríguez, and D. Youakim, "Practical formulation of obstacle avoidance in the task-priority framework for use in robotic inspection and intervention scenarios," *Robot. Auton. Syst.*, vol. 124, Feb. 2020, Art. no. 103396. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889019306104

[10] D. Youakim, P. Cieslak, A. Dornbush, A. Palomer, P. Ridao, and M. Likhachev, "Multirepresentation, multiheuristic A* search-based motion planning for a free-floating underwater vehicle-manipulator system in unknown environment," *J. Field Robot.*, vol. 37, no. 6, pp. 925–950, Sep. 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21923

[11] P. Cieślak and P. Ridao, "Adaptive admittance control in task-priority framework for contact force control in autonomous underwater floating manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 6646–6651.

[12] G. Marani and J. Yuh, *Introduction to Autonomous Manipulation: Case Study with an Underwater Robot, SAUVIM*, vol. 102. Berlin, Germany: Springer, Apr. 2014.

[13] E. Simetti, F. Wanderlingh, S. Torelli, M. Bibuli, A. Odetti, G. Bruzzone, D. L. Rizzini, J. Aleotti, G. Palli, L. Moriello, and U. Scarcia, "Autonomous underwater intervention: Experimental results of the MARIS project," *IEEE J. Ocean. Eng.*, vol. 43, no. 3, pp. 620–639, Jul. 2018.

[14] M. Prats, D. Ribas, N. Palomeras, J. C. García, V. Nannen, S. Wirth, J. J. Fernández, J. P. Beltrán, R. Campos, P. Ridao, P. J. Sanz, G. Oliver, M. Carreras, N. Gracias, R. Marín, and A. Ortiz, "Reconfigurable AUV for intervention missions: A case study on underwater object recovery," *Intell. Service Robot.*, vol. 5, no. 1, pp. 19–31, Jan. 2012.

[15] P. J. Sanz, P. Ridao, G. Oliver, G. Casalino, Y. Petillot, C. Silvestre, C. Melchiorri, and A. Turetta, "Trident an European project targeted to increase the autonomy levels for underwater intervention missions," in *Proc. OCEANS-San Diego*, 2013, pp. 1–10.

[16] E. Tuci, M. H. M. Alkilabi, and O. Akanyeti, "Cooperative object transport in multi-robot systems: A review of the state-of-the-art," *Frontiers Robot. AI*, vol. 5, pp. 185–215, May 2018.

[17] G. Casalino *et al.*, "Underwater intervention robotics: An outline of the Italian national project MARIS," *Mar. Technol. Soc. J.*, vol. 50, no. 4, pp. 98–107, Jul. 2016.

[18] E. Simetti and G. Casalino, "Manipulation and transportation with cooperative underwater vehicle manipulator systems," *IEEE J. Ocean. Eng.*, vol. 42, no. 4, pp. 782–799, Oct. 2017.

[19] R. Conti, E. Meli, A. Ridolfi, and B. Allotta, "An innovative decentralized strategy for I-AUVs cooperative manipulation tasks," *Robot. Auton. Syst.*, vol. 72, pp. 261–276, Oct. 2015.

[20] S. Heshmati-Alamdari, C. P. Bechlioulis, G. C. Karras, and K. J. Kyriakopoulos, "Decentralized impedance control for cooperative manipulation of multiple underwater vehicle manipulator systems under lean communication," May 2019, *arXiv:1905.04531*. [Online]. Available: http://arxiv.org/abs/1905.04531

[21] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics* (Modelling, Planning and Control). London, U.K.: Springer, 2009.

[22] G. Antonelli, *Underwater Robots* (Springer Tracts in Advanced Robotics), vol. 96. Cham, Switzerland: Springer, Dec. 2013.

[23] E. Simetti and G. Casalino, "A novel practical technique to integrate inequality control objectives and task transitions in priority based control," *J. Intell. Robot. Syst.*, vol. 84, nos. 1–4, pp. 877–902, Dec. 2016.

[24] S. Moe, G. Antonelli, A. R. Teel, K. Y. Pettersen, and J. Schrimpf, "Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis, and experimental results," *Frontiers Robot. AI*, vol. 3, p. 16, Apr. 2016. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2016.00016

[25] B. Siciliano and J.-J.-E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proc. 5th Int. Conf. Adv. Robot. Robots Unstruct. Environ.*, vol. 2, 1991, pp. 1211–1216.

[26] D. Ribas, N. Palomeras, P. Ridao, M. Carreras, and A. Mallios, "Girona 500 AUV: From survey to intervention," *IEEE/ASME Trans. Mechatronics*, vol. 17, no. 1, pp. 46–53, Feb. 2012.

[27] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image Vis. Comput.*, vol. 76, pp. 38–47, Aug. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885618300799

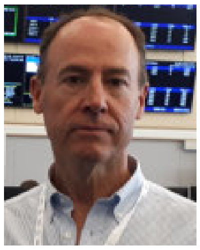
[28] N. Palomeras, A. El-Fakdi, M. Carreras, and P. Ridao, "COLA2: A control architecture for AUVs," *IEEE J. Ocean. Eng.*, vol. 37, no. 4, pp. 695–716, Oct. 2012.

[29] P. Cieslak, "Stonefish: An advanced open-source simulation tool designed for marine robotics, with a ROS interface," in *Proc. OCEANS-Marseille*, Jun. 2019, pp. 1–6.

**ROGER PI** received the B.S. degree in computer engineering from the Universitat de Girona, in 2017, the M.Sc. degree in computer vision and robotics from the University of Burgundy, the M.Sc. degree in computer vision and robotics from the Universitat de Girona, and the M.Sc. degree in computer vision and robotics from Heriot-Watt University, in 2019. He is currently pursuing the Ph.D. degree with the Underwater Robotics Research Center (CIRS) under the supervision of Dr. P. Ridao. His research interest includes autonomous motion planning for underwater vehicle manipulator systems (UVMSs). He received the Best Master Student Award from Heriot-Watt University.

**PATRYK CIEŚLAK** received the Ph.D. degree from the Department of Robotics and Mechatronics, AGH University of Science and Technology, Kraków, Poland, in 2016. During his Ph.D. studies, he was involved in several projects focused around mechatronic design and control design in mobile robotics and manipulator systems. In his Ph.D. thesis, he tackled a difficult problem of designing a new balancing mono-wheel robot, from the mechanical structure, through model-based control, to the software and firmware architecture. He is the coauthor of a commercial rehabilitation robot called Prodrobot, which is a stationary lower limbs exoskeleton, used in the learning and improvement of natural gait patterns in children. Since 2017, he has been working with the Underwater Vision and Robotics Lab (CIRS), University of Girona, Spain. He started his work at CIRS as a Marie Skłodowska-Curie Postdoctoral Fellow, after being awarded a grant for a project aimed at developing control algorithms to enable compliant control for an I-AUV performing an autonomous non-destructive testing operation. During this project, he visited Heriot-Watt University, Edinburgh, U.K., where he worked for a period of two months, in the Ocean Systems Laboratory (OSL). After the successful completion of the grant-supported project, he continued his research in underwater manipulation, tackling tasks related to obstacle avoidance, motion planning and cooperative manipulation utilising two I-AUVs. He is continuing his research at CIRS, working on underwater intervention for inspection, maintenance and repair (IMR) operations in offshore wind farms. He is also the author of an advanced open-source robot simulator called Stonefish, designed for marine robotics. This simulator is currently used in all underwater related research at CIRS and in other institutions around the world, and can be found on Github.

**PERE RIDAO** (Member, IEEE) received the Ph.D. degree in industrial engineering from the University of Girona, Spain, in 2001. He is currently the Director of the Computer Vision and Robotics Research Institute (VICOROB), the Head of the Underwater Robotics Research Center (CIRS), and an Associate Professor with the Department of Computer Engineering, University of Girona. Since 1997, he has been participated in 24 research projects (15 European and nine National), he is the author of more than 100 publications, and he has supervised nine Ph.D. theses (four more are currently under direction) and 14 M.S. theses. He is the coauthor of four licenses and one Spanish/European patent, being co-founder of Iqua Robotics S.L. spin-off company. His research interests include designing and developing autonomous underwater vehicles for 3D mapping and intervention. He has served as the Chair for the IFAC's Technical Committee on Marine Systems.



**PEDRO J. SANZ** (Senior Member, IEEE) received the B.Sc. degree in physics from the University of Valencia (UV), the M.Sc. degree in engineering (CAD/CAM) from the Technical University of Valencia (UPV), and the Ph.D. degree in computer engineering from Universitat Jaume I (UJI), Spain. He has been appointed as a Visiting Scientist with different Universities, including TUM, Germany, from 2000 to 2016, Blaise Pascal, France, in 2002, and Bologna, Italy, in 2008. He is currently a Full Professor with the Computer Science and Engineering Department, UJI, and the Head of the Interactive and Robotic Systems Lab (IRS-Lab). He has been active since the nineties in research and development, within different projects on Advanced Robotics (e.g., Coordinator of European FP7-TRIDENT project from 2010 to 2013) and this continues (e.g., Coordinator of Spanish TWINBOT project since 2018). He is the author or coauthor of a broad range of research publications. He is an active member of different scientific societies, such as IEEE (RAS, SMC, and OES), EUCog, and euRobotics. His research interests include multi-sensory based grasping and dexterous manipulation, telerobotics, and human–robot interaction (HRI), all of them applied to real life scenarios, including assistive and underwater robotics. He was a Member of the Advisory Committee of the IEEE Systems Council from 2008 to 2012 and the Humanoids Competition Chair during the "2014 IEEE-RAS International Conference on Humanoid Robots," Madrid, in 2014. He has acted as the Chair of several Tutorials and Workshops within outstanding International Conferences on Robotics (IROS, IFAC, and ICMA). He has been the Coordinator of the Spanish Robotics Network (CEA-IFAC) from 2012 to 2016. He is also the Vice-Chair of the IEEE RAS Spanish Chapter. He has served as an Associate Editor for some prestigious journals [*IEEE Robotics & Automation Magazine (RAM)* and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART C (APPLICATIONS AND REVIEWS)].